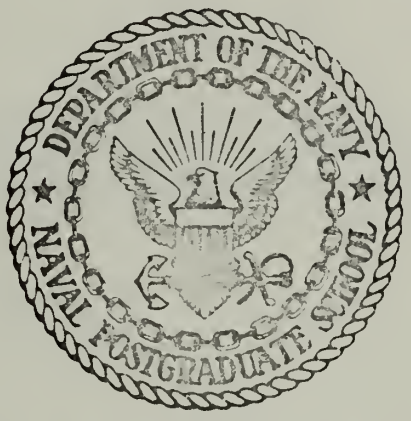
AN IMPLEMENTED TRANSFORMATIONAL SCHEME FOR MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

Robert Edward Lutz, Jr.

United States
Naval Postgraduate School



THESIS

AN IMPLEMENTED TRANSFORMATIONAL SCHEME
FOR
MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

by

Robert Edward Lutz, Jr.

Thesis Advisor: Daniel L. Davis

June 1971

Approved for public release; distribution unlimited.

An Implemented Transformational Scheme for Monic Polynomials with Integer Coefficients

by

Robert Edward Lutz, Jr.
Lieutenant, United States Navy
B.S., Auburn University, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1971

ABSTRACT

A discussion of the problem of the irreducibility of polynomials in the ring of integral polynomials establishes the framework of the research. A transformational scheme is postulated to facilitate investigation of the problem. The coherency of the scheme is detailed and the necessary computational techniques developed.

To determine the efficacy of the transformational scheme, the specification and collection of appropriate sets of data are discussed. The transformational scheme is then applied to the data and the results are tabulated and discussed.

The conclusion is drawn that a transformational scheme is a useful tool for the investigation of the irreducibility of this type of polynomial and some hypotheses as to the refinement and extension of the technique are stated.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The author wishes to express his sincere appreciation of the guidance and assistance provided by Professor Daniel L. Davis in the research for and preparation of this work. Professor Davis suggested the basic concept of a transformational scheme and contributed invaluable aid throughout the effort.

I. INTRODUCTION

A polynomial is said to be reducible in the ring of polynomials with integer coefficients, called Z[x], if it is expressible as the product of polynomials in Z[x]. A polynomial not reducible as defined above is irreducible in Z[x].

The polynomials under consideration in this research were those polynomials in Z[x] and reducibility over the ring of integers, Z. The general form of this type of polynomial will be stated as follows:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \qquad (1)$$

Investigation into the irreducibility aspects of polynomials leads to two primary avenues of research. The investigator may attempt to determine whether a given polynomial is irreducible or not. The determination is made as the result of some analysis of the polynomial. The analysis may consist of identifying some pattern among the coefficients or some predetermined relationship among the coefficients, or some other type of analysis.

The problem then becomes one of finding the appropriate specification for the given polynomial or class of polynomials which will resolve the irreducibility of the polynomial or class of polynomials under investigation. These specifications for individual polynomials or classes of polynomials are known as 'irreducibility criteria'. Several criteria are

known and apply to many different polynomials (Dorwart, 1935). However, no general irreducibility criterion has been found which will apply to all polynomials in Z[x].

Alternatively, the investigator may attempt to extract the factors of a given polynomial directly without first deciding the question of irreducibility. If the process results in the extraction of factors then the polynomial is reducible; otherwise, it is irreducible. This approach has resulted in several algorithms which extract certain types of factors (Knuth, 1969). The Kronecker method (Dorwart, 1935) is a general solution to the problem of factor extraction of polynomials in Z[x] but it is awkward to apply and the calculations involved are prohibitive in length so that the method is of little utility for practical work. An illustration of this point will be discussed later.

Thus, both paths of attack into the area of irreducible polynomials in Z[x] have been less than completely successful. A general irreducibility criterion, if one exists, has not been identified and the general solution to factor extraction is prohibitive in length even with the speed of calculation provided by the electronic computer.

II. TRANSFORMATIONAL SCHEME

After reviewing several studies in the area of irreducible polynomials in Z[x], it was decided that rather than attempt to solve the problem of irreducibility directly for a given polynomial, perhaps it would be more fruitful to attempt to facilitate the solution by somehow transforming the given polynomial into a polynomial for which the irreducibility had previously been decided, or would be easier to decide.

This decision as to a method of attack led to a search for transformations which could be used to, in some sense, simplify the polynomials under investigation. To be of any utility for this work, the transformations would have to be such that the irreducibility qualities of a polynomial would be preserved across the transformation.

The first transformation which suggested itself was to transform an arbitrary polynomial with integer coefficients, say f(x), to a primitive polynomial, i.e., a polynomial, say g(x), such that the greatest common divisor of the coefficients is unity. The irreducibility of f(x) is preserved since f(x) is the product of g(x) and some integer k, which is the gcd of the coefficients of f(x). Clearly, if f(x) is irreducible, then so is g(x) and vice versa.

Given a primitive polynomial, it may be further simplified by transforming it to a polynomial whose leading coefficient is unity, known as a monic polynomial. Consider a primitive polynomial with integer coefficients

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots\cdots + a_1 x + a_0 \qquad (2)$$

where $a_n \neq 1$. Then a transformation [3] called M, such that

$$p(x) \xrightarrow{M} a_n^{n-1}\, p\left(\frac{x}{a_n}\right) = q(x) \qquad (3)$$

and

$$q(x) = x^n + a_{n-1} x^{n-1} + a_n a_{n-2} x^{n-2} + \cdots + a_n^{n-2}\, a_1 x + a_n^{n-1}\, a_0 \qquad (4)$$

produces a monic polynomial with integer coefficients (MPWIC). The question arises as to the preservation of irreducibility across this transformation.

Assume that p(x) is irreducible in Z[x] and suppose that q(x) is reducible in Z[x]. Note that q(x) is monic, hence primitive, then

$$q(x) = k(x)\, r(x) \qquad (5)$$

$$a_n^{n-1}\, f\left(\frac{x}{a_n}\right) = k(x)\, r(x) \qquad (6)$$

$$f\left(\frac{x}{a_n}\right) = \frac{1}{a_n^{n-1}}\, k(a_n x)\, r(a_n x). \qquad (7)$$

Substituting $a_n x$ for x:

$$f(x) = \left(\frac{1}{a_n^{n-1}}\right) k(a_n x)\, r(a_n x) \qquad (8)$$

hence $f(x) = k'(x)\, r'(x)$ where $k'(x)$ and $r'(x)$ are in Q[x]. But Gauss' theorem states that f(x) is reducible in Q[x] if and only if it is reducible in Z[x]. Therefore we have a

contradiction and must conclude that if p(x) is irreducible then q(x) is also irreducible. This result establishes the preservation of irreducibility and gives the result that if p(x) is a primitive irreducible polynomial with integer coefficients then $q(x) = a_n^{n-1}\, p(x/a_n)$ is a monic irreducible polynomial with integer coefficients (MIPWIC). Since both the reduction to primitivity and the transformation to monicity preserve irreducibility, it follows that any polynomial may be transformed to a MPWIC with irreducibility properties identical to the original polynomial. Thus the only polynomials to be investigated for irreducibility are monic polynomials with integer coefficients.

Attempts were made to discover other transformations which would further simplify the MPWICS's, while preserving monicity, irreducibility, and integral coefficients. One transformation considered was

$$f(x) \longrightarrow f(x+k) \quad , \quad k \text{ in } Z \tag{9}$$

where if

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{10}$$

then

$$f(x+k) = (x+k)^n + a_{n-1}(x+k)^{n-1} + \cdots + a_1(x+k) + a_0. \tag{11}$$

Clearly, the transformation preserves monicity and integral coefficients. As to the preservation of irreducibility, suppose that f(x) is irreducible and f(x+k) is reducible. Consider $f(x+k) = g(x)\, h(x)$. Replacing x by x-k we get the expression

$$f(x) = g(x-k)\ h(x-k) \tag{12}$$

$$f(x) = g'(x)\ h'(x) \tag{13}$$

where $g'(x)$ and $h'(x)$ are in $Z[x]$. Therefore $f(x+k) = g(x)\ h(x)$ implies that $f(x) = g(x-k)\ h(x-k)$ which is a contradiction of the assumption that $f(x)$ is irreducible. Hence, irreducibility is preserved across the transformation. This transformation will be subsequently referred to as the 'T' transformation. Other notation used to describe the transformation will be

$$f(x) \xrightarrow{T_k} g(x) \quad \text{and} \quad fT_k = g. \tag{14}$$

The next transformation considered will be known as the 'D' transformation in that its effect on a polynomial is a dilation of the axes. The D transformation is specified by

$$f(x) \longrightarrow b^n\ f\left(\frac{x}{b}\right) \quad , \quad n = \deg\ (f(x)). \tag{15}$$

Using similar notation as for the T transformation, other forms are

$$f(x) \xrightarrow{D_b} g(x) \quad \text{and} \quad fD_b = g.$$

The effect of this transformation is given by the following expression. If

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 \tag{16}$$

then

$$b^n\ f\left(\frac{x}{b}\right) = x^n + a_{n-1}bx^{n-1} + \cdots + a_1b^{n-1}x + a_0b^n. \tag{17}$$

Clearly, monicity is preserved. If the number b is an integer then the transformation preserves integral coefficients. However, this number may be the reciprocal of an integer if the coefficients of f(x) are such that b^m divides a_{n-m} for each coefficient of f(x). The preservation of irreducibility is proven by methods analogous to the monicity transformation using Gauss' theorem.

A third transformation, called the X transformation, was considered. This transformation consisted of replacing the variable in f(x) by its reciprocal and multiplying the resulting polynomial by x^n. This transformation was shown to preserve integral coefficients after rationalization but preserved monicity only in the case where the last coefficient of f(x) was unity since its effect on a polynomial is to reverse the order of the coefficients. Thus the X transformation was eliminated from further consideration as being too limited in application for the purposes of this work.

Noting that each different value of k or b for the T and D transformations will produce a unique irreducible polynomial it is clear that a single irreducible polynomial generates an infinite number of irreducible polynomials. This leads to the idea of 'equivalence classes' of polynomials with an equivalence class being defined as a set of irreducible polynomials of like degree, each of which could be produced from any other member of the set through a properly chosen sequence of D or T transformations.

Several relationships were developed which relate the effects of sequential series of the two transformations.

$$f(x) \xrightarrow{T_{k_1}} f(x+k1) \xrightarrow{T_{k_2}} f(x+k1+k2) \qquad (18)$$

but, if $k' = k_1 + k_2$, then

$$f(x) \xrightarrow{T_{k'}} f(x+k') = f(x+k_1+k_2). \qquad (19)$$

Thus, the effect of any pair of successive T transformations can be produced by a single T transformation with a properly chosen value of k. Through repeated applications of Equation (19), it is clear that any finite sequence of T transformations can be replaced by a single T transformation.

Similarly,

$$f(x) \xrightarrow{D_{b_1}} b_1^n \, f\left(\frac{x}{b_1}\right) \xrightarrow{D_{b_2}} b_1^n \, b_2^n \, f\left(\frac{x}{b_1 b_2}\right) \qquad (20)$$

but, if $b_3 = b_1 b_2$, then

$$f(x) \longrightarrow b_3^n \, f\left(\frac{x}{b_3}\right) = b_1^n \, b_2^n \, f\left(\frac{x}{b_1 b_2}\right). \qquad (21)$$

Therefore, the effect of any two successive D transformations can be achieved by a single D transformation through the proper choice of the transformation factor.

Now, consider successive transformations of different types, i.e., a D transformation followed by a T transformation, or vice versa. Consider the $T_k D_b$ pair in the following expression:

$$f(x) \xrightarrow{T_k} f(x+k) \xrightarrow{D_b} b^n \, f\left(\frac{x}{b} + k\right) . \qquad (22)$$

But, after rationalization, the result is

$$f(x) \xrightarrow{D_b} b^n f\left(\frac{x}{b}\right) \xrightarrow{T_k} b^n f\left(\frac{x+k}{b}\right) . \qquad (24)$$

In this case, simple algebraic manipulation yields the following expression:

$$f(x) \xrightarrow{D_b T_k} b^n f\left(\frac{x}{b} + \frac{k}{b}\right) , \qquad (25)$$

which is the same result as would be affected by the $T_{k/b}D_b$ transformation pair. From the establishment of the two pair relationships

$$T_k D_a = D_a T_{ka} \qquad (26)$$

and

$$D_a T_k = T_{\frac{k}{a}} D_a , \qquad (27)$$

along with Equations (19) and (21), it follows that any finite sequence of D and T transformations may be collapsed into a DT or TD pair of transformations, by first collapsing the runs of like transformations and then reversing unlike pairs to produce successive like pairs which may again be collapsed into single transformations. The reduction process may be continued until the effect of the sequence is represented by, at most, a TD or DT pair of transformations.

Single transformations, either T or D, are included in the case of pairs since, from the definition of the transformations, it follows that a T_0 or D_1 transformation returns the original polynomial. Thus, any single transformation may be considered to be part of a DT or TD pair, where the other

transformation in the pair is the identity transformation for the unspecified type of transformation. Note also that each type of transformation has its inverse. The inverse of the T_k transformation is T_{-k} and the inverse of the D_a transformation is $D_{1/a}$.

Summarizing, a transformational scheme has been developed, and an algebra of the operators established, which will permit the manipulation of the polynomials under investigation in this research.

III. DEVELOPMENT OF ALGORITHMS FOR TRANSFORMATIONAL OPERATIONS

The transformational scheme described in Section II establishes a coherent framework for manipulation of polynomials. However, in order to actually perform these operations on a digital computer, a series of algorithms must be developed. Further, to be of more than nominal value, the algorithms have to be general, i.e., perform the desired operation on a monic polynomial with integer coefficients of degree n where n is a positive integer greater than or equal to 2.

The first operation to be considered is the single D transformation. By definition

$$f(x) \xrightarrow{D_b} b^n f\left(\frac{x}{b}\right) . \tag{28}$$

Let

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{29}$$

then

$$b^n f\left(\frac{x}{b}\right) = b^n \left[a_n \left(\frac{x^n}{b^n} \right) + a_{n-1} \left(\frac{x^{n-1}}{b^{n-1}} \right) + \cdots + a_1 \left(\frac{x}{b} \right) + a_0 \right] \tag{30}$$

$$= a_n x^n + a_{n-1} b x^{n-1} + \cdots + a_1 b^{n-1} x + a_0 b^n \tag{31}$$

$$= \sum_{i=0}^{n} a_{n-i} b^i x^{n-i} . \tag{32}$$

Thus, Equation (32) establishes an algorithm for computing the coefficients of $b^n f(x/b)$ given the coefficients of $f(x)$,

the degree n, and the value of b. Programming of the algorithm is straightforward and will not be discussed further.

An algorithm for computing the coefficients of the result of a T transformation was more difficult since, except for the leading coefficient, there is no simple relationship between the coefficients of the original polynomial and the transformed polynomial. The definition of the T transformation gives the expression

$$f(x) \xrightarrow{T_k} f(x+k) \quad . \tag{33}$$

Let

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{34}$$

then

$$f(x+k) = a_n (x+k)^n + a_{n-1} (x+k)^{n-1} + \cdots + a_1 (x+k) + a_0 . \tag{35}$$

For an algorithmic solution, consider Taylor's formula, as given by the expression

$$f(x) = \frac{f(a)}{0!} + \frac{f'(a)\ (x-a)}{1!} + \cdots + \frac{f^{(n)}(a)\ (x-a)^n}{n!} . \tag{36}$$

Substituting x+k for x and letting k=a, Taylor's formula becomes

$$f(x+k) = \frac{f(k)}{0!} + \frac{f'(k)\ x}{1!} + \cdots + \frac{f^{(n)}(k)\ x^n}{n!} \tag{37}$$

which yields the interesting result that the coefficient, b_i, of f(x+k) is given by

$$b_i = \frac{f^{(i)}(k)}{i!} . \tag{38}$$

Since

$$f(k) = a_n k^n + a_{n-1} k^{n-1} + a_{n-2} k^{n-2} + \cdots + a_1 k + a_0 \tag{39}$$

then

$$f'(k) = a_n n k^{n-1} + a_{n-1}(n-1)k^{n-2} + a_{n-2}(n-2)k^{n-3} + \cdots + a_1 \tag{40}$$

$$\vdots$$

$$f^{(n-m)}(k) = a_n(n)(n-1) \cdots (n-1+1)k^m + a_{n-1}(n-1)(n-2) \cdots (n-1)k^{m-1} + \cdots + a_{n-m+1} \tag{41}$$

$$\vdots$$

$$f^{(n-1)}(k) = a_n(n)(n-1) \cdots (2)\, k + a_{n-1}(n-1)(n-2) \cdots (1) \tag{42}$$

$$f^{(n)}(k) = a_n(n)(n-1) \cdots (1). \tag{43}$$

Stating the general term of the sequence in series notation gives the expression

$$f^{(n-m)}(k) = \sum_{i=0}^{m} a_{n-i} \frac{(n-i)!\, k^{m-i}}{(m-i)!} \tag{44}$$

Substituting Equation (44) into Equation (39) for $f^{(n-m)}(k)$:

$$f(x+k) = \sum_{j=0}^{n} \frac{\left(\sum_{i=0}^{j} a_{n-i} \frac{(n-i)!\, k^{j-i}}{(j-i)!} \right) x^{n-j}}{(n-j)!} \tag{45}$$

Equation (45) provides an algorithm for computing the coefficients of f(x+k) given the coefficients of f(x), the degree n of f(x), and a value of k. Again, as in the D transformation algorithm, the programming is straightforward except for the computation of the necessary factorials, which can be handled by an iterative or recursive function.

The development of the algorithm for the T and D transformations permits the transformation by computer of any polynomial by any value of k and any non-zero value of a. However, since the primary goal of the research, hence, transformational scheme, is to reduce sets of polynomials into classes rather than generate classes of polynomials, investigation of methods of determining whether a transformation exists between two arbitrary polynomials of like degree is of greater interest.

The cases of two polynomials differing by a single transformation or by a sequence of like transformations will not be discussed separately as the relationship of a single transformation to a pair of transformations was discussed in Section II and an algorithm for a pair will include the cases of single transformations. As has previously been established, any sequence of T and D transformations can readily be collapsed into a DT or TD pair and, further, a TD pair can be replaced by a DT pair of transformations and vice versa. Thus, any algorithm to determine the transformational relationship between two polynomials of like degree need only determine if they are related by, at most, a pair of unlike

transformations. Rather than test for both types of pairs an algorithm which determines only a DT or TD relationship is sufficient since the existence of one pair implies the existence of the other pair between the same two polynomials.

On examination of Equations (26) and (27), which established the relationship of the two pairs for any two polynomials, note that in the DT pair the total translation factor is either k or ka. Hence, if k and a are both integers, the total translational factor of the pair is also an integer. Also, from the definition of the D transformation, note that the factor, a, must be either an integer or the reciprocal of an integer if the coefficients of the transformed polynomial are to be integers. From the statement of the inverse of the D transformation, it is to be noted that if the D_aT_k or D_aT_{ka} transformation pair, whichever exists, from f(x) to g(x) is such that a is not an integer, then the D_aT_k or D_aT_{ka} pair, respectively, from g(x) to f(x) will be such that a is an integer. Then, since k must always be an integer, between any two polynomials which are related through a sequence of transformations, there exists a DT pair such that the T factor and the D factor are both integers. That is, of the four relationships

$$f(x) \xrightarrow{D_aT_k} g(x) \iff f(x) \xrightarrow{D_aT_k} g(x) \quad (46)$$

$$f(x) \xrightarrow{D_aT_{ka}} g(x) \iff f(x) \xrightarrow{T_kD_a} g(x) \quad (47)$$

$$g(x) \xrightarrow{D_a T_{ka}} f(x) \Longleftrightarrow g(x) \xrightarrow{T_k D_a} f(x) \quad (48)$$

$$g(x) \xrightarrow{D_a T_k} f(x) \Longleftrightarrow g(x) \xrightarrow{D_a T_k} f(x) \quad (49)$$

restated here for clarity, at least one must be such that both transformational factors are integers in the DT pair.

Hence, the algorithm has only to determine whether or not at most a DT pair with integral factors exists between two polynomials to completely decide whether they are related by any sequence of transformations.

Let

$$f(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0 \quad (50)$$

and

$$g(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0 \quad (51)$$

be two polynomials such that

$$f(x) \xrightarrow{D_a T_k} a^n f\left(\frac{x+k}{a}\right) = g(x) . \quad (52)$$

Substitution in Equation (52) gives the expression

$$a^n b_n \left(\frac{x+k}{a}\right)^n + b_{n-1} \left(\frac{x+k}{a}\right)^{n-1} + \cdots + b_1 \frac{x+k}{a} + b_0 =$$

$$c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0 . \quad (53)$$

By repeated differentiation of both sides, we obtain

$$a^n f^{(n-2)}\left(\frac{x+k}{a}\right) = a^n b_n \left(\frac{n!}{2}\right)\left(\frac{1}{a^{n-2}}\right)\left(\frac{x+k}{a}\right)^2 + b_{n-1}(n-1)! \cdot$$

$$\cdot \left(\frac{1}{a^{n-2}}\right)\left(\frac{x+k}{a}\right) + b_{n-2}(n-2)! = c_n\left(\frac{n!}{2}\right)x^2 + c_{n-1}(n-1)!\,x$$

$$+ c_{n-2}(n-2)! = g^{(n-2)}(x) \qquad (54)$$

and

$$a^n f(n-1)\left(\frac{x+k}{a}\right) = a^n b_n n!\left(\frac{1}{a^{n-1}}\right)\left(\frac{x+k}{a}\right) + b_{n-1}(n-1)!\left(\frac{1}{a^{n-1}}\right) =$$

$$c_n(n!)\,x + c_{n-1}(n-1)! = g^{(n-1)}(x) \qquad (55)$$

Since Equations (54) and (55) are true for all values of x, they are certainly true for x = 0. Simplifying and evaluating at x = 0, it follows that

$$a^2 b_n\left(\frac{n!}{2}\right)\left(\frac{k}{a}\right)^2 + a^2 b_{n-1}(n-1)!\left(\frac{k}{a}\right) + b_{n-2}(n-2)! =$$

$$c_{n-2}(n-2)! \qquad (56)$$

and

$$ab_n n'\left(\frac{k}{a}\right) + ab_{n-1}(n-1)! = c_{n-1}(n-1)!. \qquad (57)$$

Further simplification produces a system of two equations in two unknowns, k and a.

$$k^2\left[b_n n(n-1)\right] + k\left[2b_{n-1}a(n-1)\right] + \left[2b_{n-2}a^2 - 2c_{n-2}\right] = 0 \qquad (58)$$

and

$$b_n kn + b_{n-1}a = c_{n-1}\,. \qquad (59)$$

Solving Equation (59) for a in terms of k and substituting for a in Equation (58), an equation in k is produced which

is easily solvable by the use of the quadratic formula:

$$k^2\left[b_n n b_{n-1}^2(1-n) + 2b_n^2 n^2 b_{n-2}\right]$$

$$+ k\left[2c_{n-1}n(b_{n-1}^2 - 2b_{n-2}b_n - 2c_{n-1}b_{n-1}^2\right] \qquad (60)$$

$$+ \left[2b_{n-2}c_{n-1}^2 - 2b_{n-1}^2 c_{n-2}\right] = 0 \ .$$

If a solution of Equation (60) is a non-negative integer, Equation (59) is solved with this value of k. If a is also an integer, then the $D_a T_k$ transformation is performed on f(x) using the algorithms previously developed and the result is compared to g(x). If $f(x)D_a T_k = g(x)$ then there is a relationship between f(x) and g(x) and the factors of the transformation are k and a from Equations (59) and (60). If neither solution of Equation (60) is satisfactory then Equation (60) is modified by substituting b_i for c_i and c_i for b_i and the result is solved in the same manner. This interchange technique allows testing for an integral pair of transformations from g(x) to f(x). If $g(x)D_a T_k = f(x)$ for some values k and a then again there is a relationship and it is specified. If not, there exists no transformational relationship between f(x) and g(x).

The implementation of the algorithm is straightforward except for the special cases of b_{n-1} and/or c_{n-1} being zero since Equation (59) cannot be solved in these cases. The

technique utilized in these special cases is to solve Equation (59) for k instead of a and then substitute this solution into Equation (60) giving the following system of equations:

$$a^2\left[b_{n-1}(b_{n-1}-2)(n-1) + 2b_n b_{n-2} n\right]$$

$$+ a\left[2b_{n-1}(c_{n-1}-1)(n-1)\right] + \left[c_{n-1}^2(n-1) - 2b_n c_{n-2} n\right] = 0 \tag{61}$$

and

$$b_n nk + ab_{n-1} = c_{n-1} \,. \tag{62}$$

Equation (62) is always solvable since the degree n is equal to or greater than 2 and $b_n = 1$ due to the monicity constraint. Clearly, if $b_{n-1} = c_{n-1} = 0$ then $k = 0$ but this is expected since, from Equation (38),

$$b_i = \frac{f^{(i)}(k)}{i!} \tag{63}$$

hence,

$$c_{n-1} = b_n\left(\frac{n!}{2}\right) k + b_{n-1}(n-1)! \,. \tag{64}$$

Thus, since $b_n = 1$, the condition $k = 0$ is the only one which, along with the prior condition that $a_{n-1} = 0$, will allow $b_{n-1} = 0$. Therefore, if both a_{n-1} and b_{n-1} are equal to zero, the transformations, if any, between f(x) and g(x) must be solely of the D type since $k = 0$. This possibility is tested by solving for a D transformation factor from b_0

and c_0. That is, if $g(x) = a^n f(x/a)$ then $c_0 = a^n b_0$. Solving for integral values of a and then calculating the trial polynomial as in the general case allows the testing of this condition.

With the development of the algorithm for testing f(x) and g(x) pairs, the necessary algorithms have been developed for performing either the D or T transformations on a given polynomial or determining the transformational relationship between two polynomials.

IV. SPECIFICATION AND COLLECTION OF DATA BASE

To test the utility and efficacy of the developed transformational scheme and the associated algorithms as a technique for investigation of irreducible polynomials in Z[x], a representative data base of irreducible polynomials was necessary. To provide a large but finite number of polynomials from which to select the irreducible polynomials, certain conditions were specified and all polynomials meeting these conditions were selected and then screened for irreducibility. The conditions for selection were as follows:

(a) f(x) is a monic polynomial with integer coefficients

(b) f(k) is in Z for k in Z

(c) the absolute value of f(k) is less than or equal to 5 for k = 0,1,...,n where n is the degree of f(x).

Conditions (b) and (c) imply a grid of points in the x-y plane, which is represented graphically in Fig 1. The data base is the set of all polynomials satisfying the three conditions specified above. Note that monicity, by definition, requires that the leading coefficient, a_n, be equal to 1, rather than $|a_n| = 1$. But if a polynomial, say f(x), exists such that it satisfies all the conditions for inclusion except that $a_n = -1$, then there exists another polynomial say g(x), such that f(x) = -g(x) and g(x) will be included in the data base. That is, each of the monic polynomials included in the data set will have a reflection across the

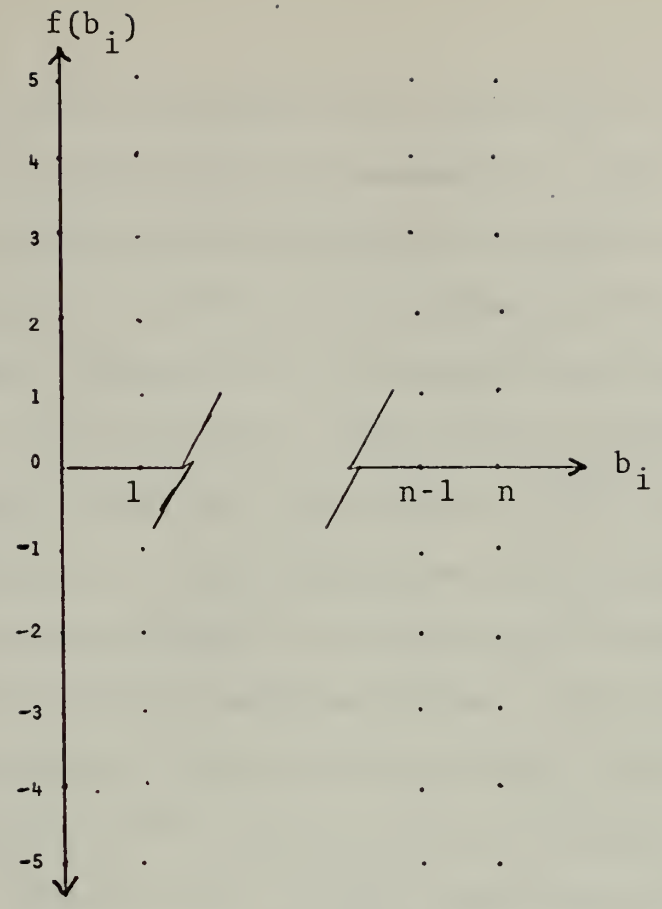


Figure 1.

x-axis which lies on the degree grid but is not included in the data base. But the factorization or irreducibility of the reflected polynomial is identical to that of its reflection with the additional factor of -1. Hence, the irreducibility or factorization of both a polynomial, $f(x)$, and its reflection, $-f(x)$, are represented by the inclusion of $f(x)$ in the data base.

The actual calculation of all the polynomials for a given degree grid can be accomplished by use of the LaGrange interpolation formula

$$f(x) = \sum_{i=0}^{n} f(b_i) \frac{(x-b_0)\ldots(x-b_{i-1})(x-b_{i+1})\ldots(x-b_n)}{(b_i-b_1)\ldots(b_i-b_{i-1})(b_i-b_{i+1})\ldots(b_i-b_n)} \quad (65)$$

where b_i are the abcissae and the $f(b_i)$ are the ordinates of the n+1 points for the summation. After the calculation, the polynomial can be screened for monicity and integral coefficients. Clearly, this technique requires that a LaGrangian polynomial be calculated for every possible set of $f(b_i)$ over the grid. Note that the number of sets of functional values has the order of 11^{n+1} where n is the maximum degree of the LaGrangian polynomials. Hence, the number of polynomials to be calculated becomes extremely large for grids of moderate degree. The calculations for each polynomial consist of the actual calculation of the polynomial and then testing the polynomial for monicity and integral coefficients. After programming the LaGrange formula and executing test runs with limited grids, it was determined that approximately 780 microseconds were required for each fifth degree polynomial. Extrapolating to the full grid for the fifth degree, this technique would have required just under 24 hours of central processing unit time on an IBM 360/67 for the processing of the 1,771,561 possible sets of points. Thus an exhaustive search technique, which is analagous to the Kronecker method of factorization, is prohibitive in terms of time required.

Examination of the grid and LaGrangian formula, along with the statement of the problem, offers several areas for reduction of the number of polynomials to be calculated. The previously mentioned case of reflected pairs allows elimination of half the sets of functional values. Thus, if the

the condition that $f(0) \geq 0$ is imposed then one member of each reflected pair is eliminated and the number of polynomials to calculate is halved. Additionally, since the goal is to find irreducible polynomials, any set of points such that at least one $f(b_i) = 0$ may be eliminated since for any k, $f(k) = 0$ implies that k is a root and hence $(x-k)$ is a factor of $f(x)$. Therefore, $f(x)$ is reducible.

Considering the LaGrange formula and actual calculation of the polynomials, note that the requirement that the polynomials be monic imposes constraints on the possible values of $f(b_i)$. The LaGrange formula may be written in the form

$$f(x) = \sum_{i=0}^{n} f(k) \frac{Y_k(x)}{Y_k(k)} \tag{66}$$

where

$$Y_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} (x-j). \tag{67}$$

Then

$$f^{(n)}(0) = \sum_{k=0}^{n} f(k) \frac{Y_k^{(n)}(0)}{Y_k(k)} \tag{68}$$

but, since Y_k is monic by construction

$$Y_k^{(n)}(0) = n! \tag{69}$$

hence

$$f^{(n)}(0) = \sum_{k=0}^{n} f(k) \frac{n!}{Y_k(k)} . \tag{70}$$

By definition, $Y_k(k) = k(k-1)\ldots(1)(-1)\ldots(-(n-k))$ thus

$$Y_k(k) = k!\ (-1)^{n-k}\ (n-k)! \quad . \tag{71}$$

Substituting Equation (71) into Equation (70), we have the expression

$$f^{(n)}(0) = \sum_{k=0}^{n} f(k) \frac{n!}{k!\ (-1)^{n-k}\ (n-k)!} \tag{72}$$

or

$$f^{(n)}(0) = \sum_{k=0}^{n} f(k) \left\{ \begin{matrix} n \\ k \end{matrix} \right\} (-1)^{n+k} \tag{73}$$

But $f^{(n)}(0) = f^{(n)}(x)$ which is just the leading coefficient of $f(x)$. Since the goal is to find monic polynomials then only those sets of points which satisfy

$$1 = \sum_{k=0}^{n} f(k) \left\{ \begin{matrix} n \\ k \end{matrix} \right\} (-1)^{n+k} \tag{74}$$

can possibly be functional values of a monic polynomial. Thus, only sets of points which satisfy Equation (74) need be considered and the LaGrangian polynomials need be calculated for only these sets of points.

In order to illustrate the savings effected through the use of the condition established by Equation (74), consider

the 1408 minutes of CPU time necessary to calculate and test all the sets for the fifth degree grid. Elimination of the reflected pairs sets and those sets containing a zero still leaves 500,000 sets of points to consider which would require approximately 398 minutes of CPU time. The conditions of $f(b_i) \neq 0$ and Equation (74) were programmed and these 500,000 sets were tested. In this case 551 sets were found in approximately 38 seconds of CPU time. The 551 LaGrange polynomials were then calculated and tested in another 32.5 seconds for a total of 70.5 seconds compared to the 398 minutes required for the exhaustive search technique described above.

The same conditions were applied to the grids for degree 2,3, and 4 and a limit of ±5 for each case. The results are summarized below:

TABLE I

LIMIT X DEGREE GRID	TOTAL SETS	SETS SATISFYING EQUATION (74) AND $f(b_i) \neq 0$	NUMBER OF MPWIC'S FROM LA-GRANGE FORMULA
5 X 2	500	44	44
5 X 3	5000	253	132
5 X 4	50000	1067	107
5 X 5	500000	551	0

The monic polynomials calculated for each grid (see Appendix A) are a superset of the irreducible MPWIC's for

each grid since they are the set of all the monic polynomials with integer coefficients except those which contain a linear factor of $(x-b_i)$ since those polynomials have already been shown to be reducible. However, polynomials with linear factors of the form $(x+k)$ have not been eliminated. Thus each set of polynomials was further screened by a synthetic division program to eliminate those polynomials which contained this type of linear factor. Since the absolute value of the constant term of any linear factor of a polynomial must be less than or equal to the absolute value of the constant term of the polynomial if the polynomial is reducible in Z[x], the synthetic division program tested those values for each polynomial under test. This test, combined with the previous conditions imposed upon the sets of polynomials, constituted a sufficient condition for irreducibility for the MPWIC's of degree 2 and 3 since if MPWIC's of these degrees are reducible in Z[x], they must contain a linear factor. However, those polynomials of degrees $\geq$ 4 may not contain a linear factor and still be reducible in Z[x] as the product of two irreducible polynomials of degree $\geq$ 2.

Suppose that

$$f(x) = g(x)\, h(x) \tag{75}$$

then for any value k, f(k) is the product of g(x) and h(x), i.e., f(k) is divisible by g(k) and h(k) for any value of k. If f(x), g(x), and h(x) are all in Z[x], then for an integral

value of k, f(k), g(k), and h(k) are in Z. Also, if $f(k) \neq 0$, then $g(k) \neq 0$ and $h(k) \neq 0$. Now if an integer is the product of two or more integers, none of which is zero, then the absolute value of each of the factors of the product is less than or equal to that of the product. That is, if $f(x) = g(x)\, h(x)$ then for any value of x

$$|f(x)| \geq |g(x)| \geq 1 \tag{76}$$

and

$$|f(x)| \geq |h(x)| \geq 1 \ . \tag{77}$$

Applying these results to the polynomials under discussion, it follows that if f(x) is a monic polynomial with integer coefficients of degree ≥ 4, $|f(k)| \leq 5$, and f(x) has no linear factors but is reducible in Z[x], then its factors must be polynomials over a grid of lesser degree with bound equal to 5. In the case of the monic polynomials, f(x), of degree 4 with no linear factors and such that $|f(k)| \leq 5$ for k = 0,1,..,4 then any possible irreducible quadratic factors must be included in the set of irreducible quadratics already determined. Thus, trial division of the quartics with no linear factors by each of the irreducible quadratics will establish the irreducibility of the quartic in question. The actual technique employed was the testing of each coefficient of the quartic successively during the division so that the full division occurred only if the quadratic was actually a factor of the quartic.

The results of the various irreducibility techniques are summarized as follows:

TABLE II.

LIMIT X DEGREE GRID	NUMBER OF MPWIC's	NUMBER WITH NO LINEAR FACTOR	NUMBER OF IRREDUCIBLE NPWIC's
5 X 2	44	43	43
5 X 3	132	126	126
5 X 4	107	107	96

The results as illustrated in Table II complete the establishment of data bases of irreducible polynomials for three separate sets of polynomials upon which the utility of the developed transformational scheme and algorithms can be tested. The irreducible polynomials for each grid are listed in Appendix A.

V. APPLICATION OF TRANSFORMATIONAL SCHEME TO DATA BASE

In order to acquire knowledge of the effect of the transformational scheme on a given data base and hence arrive at some conclusions as to the utility of the scheme, the transformational scheme was applied to the data bases described in Section IV, using a program called "TESTER" which actually implements the testing of polynomial pairs as described in Section III. The execution of TESTER actually separates a set of polynomials into equivalence classes, each of which consists of all the polynomials of the original set which are related to each other through the transformation scheme. The application to the sets of polynomials resulted in the reduction of the 43 quadratic irreducible MPWIC's into 13 equivalence classes, 126 cubic irreducible MPWIC's into 91 equivalence classes, and 96 quartic MPWIC's into 93 equivalence classes. These classes are tabulated in Appendix B.

As a check on these results, an interactive program called "XFORM" was written and executed which actually performs desired sequences of D and T transformations upon a selected polynomial. Independently of the results of "TESTER" the same data bases were manipulated by the use of XFORM and the same results were achieved through trial and error transformation sequences.

Also an interactive program called TESTERA was written which is merely a minor revision of TESTER in that two

polynomials are entered via an interactive terminal and TESTERA determines whether or not a transformational relationship exists and if so, describes the relationship.

VI. RESULTS, CONCLUSIONS, AND HYPOTHESES

This research has shown that any polynomial which, when transformed to a monic polynomial, has integer coefficients and is coincident to the grid for its degree over the range of abcissae of the grid, can be tested for irreducibility by testing it for a transformational relationship to one member of each of the classes of irreducible classes of irreducible MPWIC's for that particular degree grid. If a relationship exists with any of these classes then the polynomial is irreducible. Note that the polynomial is tested for irreducibility and is found to be reducible by default rather than the usual technique of attempting to determine reducibility and finding irreducibility by default.

Since there are infinitely many polynomials which may be tested, the primary result of this research has been the establishment of an irreducibility criterion which can be easily and quickly applied to an infinite set of polynomials. For instance, any quartic monic polynomial, say $f(x)$, such that $|f(x)| \leq 5$ for any 5 consecutive integer arguments can be tested for irreducibility by merely transforming the polynomial by means of the T transformation so that the 5 arguments are coincident to the abcissae of the degree 4, limit 5 grid.

It must be noted that, due to the severely limited grids selected, the utility of the D transformation is

demonstrable only for the data base for the quadratics. That is, the limit constraint of $|f(x)| \leq 5$ precluded the possibility of the existence of $f(x)$ and $g(x)$ such that $g(x) = a^n f(x/a)$ and both $f(x)$ and $g(x)$ being coincident to the grid for degree greater than or equal to three. However, in the quadratic case, this limit restriction does not preclude the possibility and results demonstrating the effectiveness of the D transformation were achieved. The execution of a modified version of TESTER which tested only for T transformational relationships between members of the set of 43 irreducible quadratics resulted in the establishment of 16 equivalence classes under this transformation. As previously noted, the testing for DT pairs of transformations resulted in the establishment of only 13 equivalence classes. Comparison shows that 3 pairs of classes under the T transformation are compressed into 3 classes under the DT transformational pair. Thus, the utilization of the D transformation along with the T transformation extended the capabilities of the techniques.

In view of the results of this work, further research into the technique would seem warranted. This further research, if undertaken, could be to extend the technique in one or more of several possible directions. Perhaps other and more powerful transformations could be identified and included in the transformational scheme. Or modification of the specifications of the various data sets might prove fruitful as this author has identified no conceptual barrier to either extension. Merely relaxing the limit constraint

to some other fixed value, possibly the sum of the degree and some well-chosen constant, should provide interesting results when compared to the results of this work. Alternatively, some variable limit such as $f(x) = x^n$ where n is the degree of f(x) might provide some additional insight into the total problem of a transformational scheme of irreducibility criteria.

APPENDIX A

TABLE 1

COEFFICIENTS OF IRREDUCIBLE QUADRATIC MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID

COEFFICIENTS LISTED AS : AX2 + BX + C

A	B	C	A	B	C	A	B	C
1	-5	1	1	-4	1	1	1	-1
1	-3	1	1	-1	-1	1	-1	1
1	-2	-1	1	0	1	1	-3	-1
1	-4	-1	1	-5	2	1	-4	2
1	0	-2	1	-2	2	1	-1	2
1	-2	-2	1	-3	-2	1	-6	3
1	-5	3	1	1	-3	1	-3	3
1	0	-3	1	-2	3	1	-1	-3
1	-1	3	1	-3	-3	1	-6	4
1	2	-4	1	1	-4	1	-3	4
1	-2	4	1	-1	-4	1	-2	-4
1	-7	5	1	3	-5	1	-5	5
1	2	-5	1	-4	5	1	1	-5
1	-3	5	1	0	-5	1	-2	5
1	-1	-5						

TABLE 2

COEFFICIENTS OF IRREDUCIBLE CUBIC MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID
COEFFICIENTS LISTED AS : AX3 + BX2 + CX + D

A	B	C	D	A	B	C	D	A	B	C	D
1	-6	11	-1	1	-7	12	-1	1	-1	-5	1
1	-6	10	-1	1	-7	11	-1	1	-2	-3	1
1	-5	8	-1	1	-6	9	-1	1	-3	-1	1
1	-5	7	-1	1	-6	8	-1	1	-4	1	1
1	-3	0	1	1	-4	5	-1	1	-5	6	-1
1	-5	4	1	1	-4	3	1	1	-3	2	-1
1	-4	3	-1	1	-5	5	1	1	-4	4	1
1	-3	1	-1	1	-4	2	-1	1	-6	7	1
1	-5	6	1	1	-2	-1	-1	1	-3	0	-1
1	-6	8	1	1	-5	7	1	1	-2	-2	-1
1	-7	10	1	1	-6	9	1	1	-1	-4	-1
1	-7	13	-2	1	-8	14	-2	1	-6	11	-2
1	-7	12	-2	1	-6	10	-2	1	-2	-3	2
1	-5	8	-2	1	-3	-1	2	1	-2	-2	2
1	-6	8	-2	1	-4	2	2	1	-4	4	-2
1	-5	5	-2	1	-5	4	2	1	-3	2	-2
1	-4	3	-2	1	-4	4	2	1	-3	1	-2
1	-5	6	2	1	-2	-1	-2	1	-6	8	2
1	-5	7	2	1	-7	14	-3	1	-8	15	-3
1	-1	-6	3	1	-6	11	-3	1	-7	12	-3
1	-3	-2	3	1	-2	-3	3	1	-5	8	-3
1	-6	9	-3	1	-4	1	3	1	-3	0	3
1	-4	5	-3	1	-5	6	-3	1	-5	4	3
1	-4	3	3	1	-3	2	-3	1	-4	3	-3
1	-6	7	3	1	-5	6	3	1	-2	-1	-3
1	-8	16	-4	1	-7	14	-4	1	-8	15	-4
1	-6	12	-4	1	-1	-6	4	1	-6	11	-4
1	-2	-4	4	1	-5	9	-4	1	-4	0	4
1	-3	-1	4	1	-5	7	-4	1	-4	1	4
1	-4	5	-4	1	-5	6	-4	1	-3	3	-4
1	-4	4	-4	1	-4	3	4	1	-3	2	-4
1	-5	5	4	1	-2	0	-4	1	-8	17	-5
1	-9	18	-5	1	-7	15	-5	1	0	-9	5
1	-7	14	-5	1	-8	15	-5	1	-1	-7	5
1	-6	12	-5	1	-7	13	-5	1	-2	-5	5
1	-1	-6	5	1	-6	11	-5	1	-7	12	-5
1	-3	-2	5	1	-2	-3	5	1	-5	8	-5
1	-6	9	-5	1	-3	-1	5	1	-4	6	-5
1	-5	7	-5	1	-4	1	5	1	-3	0	5
1	-4	5	-5	1	-5	6	-5	1	-5	3	5
1	-4	2	5	1	-3	3	-5	1	-4	4	-5
1	-5	4	5	1	-4	3	5	1	-3	2	-5

TABLE 3

COEFFICIENTS OF IRREDUCIBLE QUARTIC MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID

COEFFICIENTS LISTED AS : AX4 + BX3 + CX2 + DX + E

A	B	C	D	E	A	B	C	D	E
1	-8	21	-20	1	1	-9	26	-24	1
1	-8	21	-19	1	1	-9	26	-23	1
1	-7	13	-3	-1	1	-8	20	-17	1
1	-9	25	-21	1	1	-7	13	-4	-1
1	-8	20	-16	1	1	-9	25	-20	1
1	-8	18	-9	-1	1	-7	15	-11	1
1	-8	20	-15	1	1	-8	19	-12	-1
1	-7	14	-8	1	1	-8	19	-12	1
1	-9	24	-17	-1	1	-8	19	-13	-1
1	-9	25	-19	-1	1	-8	20	-15	-1
1	-7	13	-5	1	1	-9	25	-20	-1
1	-8	20	-16	-1	1	-7	13	-4	1
1	-10	30	-25	-1	1	-9	25	-21	-1
1	-6	8	1	1	1	-6	7	5	-2
1	-10	31	-29	2	1	-8	21	-20	2
1	-9	26	-24	2	1	-7	13	-3	-2
1	-9	25	-21	2	1	-7	13	-4	-2
1	-8	20	-16	2	1	-8	19	-11	-2
1	-8	19	-13	2	1	-8	19	-12	-2
1	-7	14	-8	2	1	-9	25	-19	-2
1	-7	13	-5	2	1	-9	25	-20	-2
1	-8	20	-16	-2	1	-9	26	-26	3
1	-10	31	-30	3	1	-7	12	0	-3
1	-8	21	-20	3	1	-9	26	-24	3
1	-7	13	-3	-3	1	-8	20	-17	3
1	-9	25	-21	3	1	-8	19	-10	-3
1	-7	14	-6	-3	1	-9	24	-16	-3
1	-8	19	-12	-3	1	-7	14	-8	3
1	-9	25	-19	-3	1	-8	20	-15	-3
1	-7	13	-5	3	1	-8	22	-24	4
1	-6	7	6	-4	1	-10	31	-30	4
1	-7	12	0	-4	1	-9	26	-24	4
1	-7	13	-2	-4	1	-9	25	-22	4
1	-8	18	-8	-4	1	-8	19	-10	-4
1	-8	19	-14	4	1	-9	24	-16	-4
1	-7	14	-8	4	1	-9	25	-18	-4
1	-7	13	-6	4	1	-6	6	9	-5
1	-9	27	-29	5	1	-10	32	-33	5
1	-9	27	-28	5	1	-6	7	6	-5
1	-9	26	-26	5	1	-7	12	1	-5
1	-8	21	-21	5	1	-7	12	0	-5
1	-9	26	-24	5	1	-8	18	-7	-5
1	-7	13	-3	-5	1	-8	20	-17	5
1	-8	18	-8	-5	1	-7	15	-12	5
1	-8	20	-16	5	1	-7	14	-6	-5
1	-8	19	-14	5	1	-8	19	-11	-5
1	-7	14	-9	5	1	-9	24	-16	-5
1	-8	19	-12	-5	1	-7	14	-8	5

APPENDIX B

TABLE 1

EQUIVALENCE CLASSES OF QUADRATIC IRREDUCIBLE MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C
**********		
1	-5	1
1	-3	-3
1	1	-5
1	-1	-5
**********		
1	-4	1
1	-2	-2
1	0	-3
**********		
1	1	-1
1	-3	1
1	-1	-1
1	-4	-1
1	-6	4
1	2	-4
1	-2	-4
1	-5	5
1	0	-5
**********		
1	-1	1
1	-3	3
1	-2	4
**********		
1	-2	-1
1	-4	2
1	0	-2
**********		
1	0	1
1	-2	2
1	-4	5
1	-2	5
**********		
1	-3	-1
1	-5	3
1	1	-3
1	-1	-3
**********		
1	-5	2
1	-3	-2
1	1	-4
1	-1	-4
**********		
1	-1	2
1	-3	4
**********		
1	-6	3
1	2	-5
**********		
1	-1	3
1	-3	5
**********		
1	-7	5
1	3	-5
**********		
1	-2	3
**********		

TABLE 2

EQUIVALENCE CLASSES OF CUBIC IRREDUCIBLE MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C	D
1	-7	12	-1
1	-4	1	5
1	-1	-5	1
1	-7	11	-1
1	-4	0	4
1	-2	-3	1
1	-5	4	1
1	-6	9	-1
1	-3	0	3
1	-3	-1	1
1	-6	8	-2
1	-6	8	-1
1	-3	-1	2
1	-4	1	1
1	-1	-4	-1
1	-7	12	-5
1	-3	0	1
1	-6	9	-3
1	-5	5	1
1	-2	-2	2
1	-6	7	1
1	-3	-2	3
1	-2	-1	-1
1	-5	6	-3
1	-3	0	-1
1	-6	9	-5
1	-6	8	1
1	-3	-1	4
1	-2	-2	-1
1	-5	5	-2
1	-6	9	1
1	-3	0	5
1	-7	13	-2
1	-4	2	5
1	-7	12	-2
1	-4	1	4

```
 1          -2          -3           2
 1          -5           4           2
**********************************************
 1          -4           2           2
 1          -7          13          -5
**********************************************
 1          -2          -1          -2
 1          -5           6          -4
**********************************************
 1          -6           8           2
 1          -3          -1           5
**********************************************
 1          -7          14          -3
 1          -4           3           5
**********************************************
 1          -7          12          -3
 1          -4           1           3
**********************************************
 1          -2          -3           3
 1          -5           4           3
**********************************************
 1          -4           3           3
 1          -7          14          -5
**********************************************
 1          -6           7           3
 1          -3          -2           5
**********************************************
 1          -2          -1          -3
 1          -5           6          -5
**********************************************
 1          -8          16          -4
 1          -2          -4           4
 1          -5           3           5
**********************************************
 1          -7          14          -4
 1          -4           3           4
**********************************************
 1          -5           9          -4
 1          -4           6          -5
**********************************************
 1          -8          17          -5
 1          -2          -3           5
 1          -5           4           5
**********************************************
```

ALL OTHER CUBIC MIPWIC'S FROM TABLE 2 IN APPENDIX A ARE THE SINGLE MEMBERS OF THEIR RESPECTIVE EQUIVALENCE CLASSES

TABLE 3

EQUIVALENCE CLASSES OF QUARTIC IRREDUCIBLE MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C	D	E
**********	**********	**********	**********	**********
1	-10	30	-25	-1
1	-6	6	9	-5
**********	**********	**********	**********	**********
1	-10	31	-30	3
1	-6	7	6	-5
**********	**********	**********	**********	**********
1	-6	7	6	-4
1	-10	31	-30	4
**********	**********	**********	**********	**********

ALL OTHER QUARTIC MIPWIC'S FROM TABLE 3 IN APPENDIX A ARE THE SINGLE MEMBERS OF THEIR RESPECTIVE EQUIVALENCE CLASSES

NOTES AND INSTRUCTIONS ON UTILIZATION OF PROGRAM "XFORM"

1. "XFORM" is a program which performs selected "T" and/or "D" transformations on a desired polynomial.
2. The degree of the polynomial is specified by the entry of a positive integer.
3. The coefficients of the desired polynomial are entered as integers separated by commas, representing the coefficients of the polynomial in descending powers of the variable. Zero coefficients must be included.
4. The type of transformation to be performed is specified by entry of "T" or "D."
5. The translation factor is entered as an integer.
6. The dilation transformation is possible with an integer factor or a factor which is the reciprocal of an integer for certain polynomials. Thus, when the "D" transformation is selected, integer or reciprocal factors must be specified by a response of "I" or "R." Then the actual factor is specified by the entry of an integer value. If the factor is to be an integer, the value is the value entered. If the factor is to be a reciprocal the value is the reciprocal of the integer value entered.
7. The resulting polynomial is displayed as a series of integers separated by commas. These integers are the coefficients of the descending powers of the variable.
8. The "NEXT OPTION?" data entry point allows the operator to select the result of the last transformation, the base

of the last transformation, the original polynomial for the present series of transformations, or the entry of a new polynomial as the next polynomial to be transformed by means of the appropriate response. The responses are "R," "P," "0," and "" respectively.

9. Arithmetic data in the program is represented in binary integers of 31 digits. This allows a range of integers from -262,144 to 262,143 for any single value. This restriction is a limit of the IBM System/360 PL/1 implementation.

10. The output coefficients are displayed to the operator as integers separated by commas. Internally, this is a single character string constructed through various string-handling operations. The maximum length of this string is 72 characters including the commas, again due to the System /360 PL/1 implementations.

11. The maximum number of coefficients for display as output is 64 due to problem data attributes in the program. However, this restriction is insignificant in that the number of coefficients is limited to 36 from Note #9 above.

```
load xform (xeq)
EXECUTION BEGINS...
DEGREE?
_5
COEFFICIENTS SEPARATED BY COMMAS
_1,-1,1,-1,-1,1
TRANSFORMATION TYPE?
_t
K =
_2
1,9,33,61,55,19
NEXT OPTION?
_result
TRANSFORMATION TYPE?
_t
K =
_-2
1,-1,1,-1,-1,1
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_integer
A =
_2
1,-2,4,-8,-16,32
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_r
A =
_2
1,-1,1,-1,-1,1
NEXT OPTION?
_
DEGREE?
_4
COEFFICIENTS SEPARATED BY COMMAS
_1,1,1,1,1
TRANSFORMATION TYPE?
_t
K =
_2
1,9,31,49,31
NEXT OPTION?
_(continued on next page)¢
```

```
r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_2
1,18,124,392,496
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_r
A =
_2
1,9,31,49,31
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_t
K =
_-2
1,1,1,1,1
NEXT OPTION?
_
DEGREE?
_3
COEFFICIENTS SEPARATED BY COMMAS
_1,0,0,1
*TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_3
1,0,0,27
NEXT OPTION?
_o
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_-4
1,0,0,-64
NEXT OPTION?
_
DEGREE?
_
END OF JOB
R; T=2.11/6.13 13.21.40
```

```
r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_2
1,18,124,392,496
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_r
A =
_2
1,9,31,49,31
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_t
K =
_-2
1,1,1,1,1
NEXT OPTION?
_
DEGREE?
_3
COEFFICIENTS SEPARATED BY COMMAS
_1,0,0,1
*TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_3
1,0,0,27
NEXT OPTION?
_o
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_i
A =
_-4
1,0,0,-64
NEXT OPTION?
_
DEGREE?
_
END OF JOB
R; T=2.11/6.13 13.21.40
```

```
XFORM:PROC OPTIONS (MAIN);                                                    XF000010
                                                                              XF000020
  /* THIS IS A PROGRAM WHICH APPLIES VARIOUS TRANSFORMATIONS                 XF000030
      TO A POLYNOMIAL AND ALLOWS EXAMINATION OF THE EFFECT OF THESE           XF000040
      TRANSFORMATIONS.   */                                                   XF000050
                                                                              XF000060
  /* DECLARE NECESSARY VARIABLES  */                                          XF000070
                                                                              XF000080
DCL BUFFER CHAR(80),                                                          XF000090
    RESPONSE CHAR(512) VARYING,                                               XF000100
    TEMP CHAR(72) VARYING,                                                    XF000110
    (AMOUNT,DEGREE,TERM, F,I,J,K,L,M,N) FIXED BIN (31),                       XF000120
    NUMBER CHAR(16) VARYING,                                                  XF000130
    CHOICE CHAR(12) VARYING,                                                  XF000140
    DIGITS CHAR(10) INITIAL('0123456789'),                                    XF000150
    DIGITS1 CHAR(11) INITIAL('0123456789-'),                                  XF000160
    STRING1 CHAR(12) INITIAL ('0123456789,-'),                                XF000170
    ERROR CHAR(24) INITIAL ('INVALID ENTRY----REENTER');                      XF000180
                                                                              XF000190
                                                                              XF000200
  /* REQUEST DESIRED DEGREE AND PROCESS RESPONSE  */                          XF000210
                                                                              XF000220
TOP:DISPLAY('DEGREE?') REPLY (BUFFER);                                        XF000230
RESPONSE='';                                                                  XF000240
DO L = 1 TO 80;                                                               XF000250
 IF SUBSTR(BUFFER,L,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);       XF000260
END;                                                                          XF000270
IF RESPONSE = '' THEN GO TO FINISH;                                           XF000280
DO M = 1 TO LENGTH(RESPONSE);                                                 XF000290
    IF INDEX(DIGITS,SUBSTR(RESPONSE,M,1))=0 THEN DO;                          XF000300
                                                    DISPLAY (ERROR);          XF000310
                                                    GO TO TOP;                XF000320
                                                END;                          XF000330
END;                                                                          XF000340
DEGREE = RESPONSE;                                                            XF000350
                                                                              XF000360
BEGIN;                                                                        XF000370
                                                                              XF000380
  /* SET UP NECESSARY ARRAYS */                                               XF000390
                                                                              XF000400
DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));                     XF000410
DCL (TRANSP(0:DEGREE),COEFFS(0:DEGREE),POLY(0:DEGREE))                        XF000420
       FIXED BIN (31);                                                        XF000430
                                                                              XF000440
  /* REQUEST COEFFICIENTS AND PLACE RESPONSES INTO PROPER ARRAYS. */          XF000450
                                                                              XF000460
  DATA:DISPLAY('COEFFICIENTS SEPARATED BY COMMAS')                            XF000470
     REPLY(BUFFER);                                                           XF000480
```

```
RESPONSE = '';                                                                  XF000490
DO L = 1 TO 80;                                                                 XF000500
IF SUBSTR(BUFFER,L,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);          XF000510
END;                                                                            XF000520
DO L = 1 TO LENGTH(RESPONSE);                                                   XF000530
IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;                               XF000540
        DISPLAY(SUBSTR(RESPONSE,L,1));                                          XF000550
        DISPLAY(ERROR);                                                         XF000560
        GO TO DATA;                                                             XF000570
        END;                                                                    XF000580
END;                                                                            XF000590
DO L = 0 TO DEGREE ;                                                            XF000600
 IF RESPONSE = '' THEN DO; DISPLAY(ERROR); GO TO DATA; END;                     XF000610
 K= INDEX(RESPONSE,',');                                                        XF000620
 IF K=1 THEN DO;DISPLAY(RESPONSE); DISPLAY(ERROR);GO TO DATA; END;              XF000630
 IF K = 0 THEN K = LENGTH(RESPONSE) + 1;                                        XF000640
 COEFFS(L) = SUBSTR(RESPONSE,1,K-1);                                            XF000650
 IF INDEX(RESPONSE,',')=0 THEN RESPONSE='';                                     XF000660
 IF LENGTH(RESPONSE)>K THEN RESPONSE=SUBSTR(RESPONSE,K+1);                      XF000670
        ELSE RESPONSE='';                                                       XF000680
END;                                                                            XF000690
POLY=COEFFS;                                                                    XF000700
                                                                                XF000710
  /* DETERMINE TYPE OF TRANSFORMATION TO BE PERFORMED */                        XF000720
                                                                                XF000730
TRANS:DISPLAY('TRANSFORMATION TYPE?') REPLY (RESPONSE);                         XF000740
IF RESPONSE = '' THEN GO TO TOP;                                                XF000750
RESPONSE = SUBSTR(RESPONSE,1,1);                                                XF000760
IF RESPONSE¬='T' & RESPONSE¬='D' & RESPONSE¬='M' THEN DO;                        XF000770
        DISPLAY(RESPONSE); DISPLAY(ERROR); GO TO TRANS; END;                    XF000780
IF RESPONSE = 'T' THEN  CALL TRANSL;                                            XF000790
IF RESPONSE = 'D' THEN  CALL DILATE;                                            XF000800
IF RESPONSE = 'M' THEN CALL MONIC;                                              XF000810
                                                                                XF000820
TRANSL:PROCEDURE;                                                               XF000830
                                                                                XF000840
  /* THE TRANSL PROCEDURE PERFORMS THE TRANSLATION TRANSFORMATION */            XF000850
                                                                                XF000860
  /* DETERMINE THE AMOUNT OF TRANSLATION */                                     XF000870
                                                                                XF000880
TEMP = '';                                                                      XF000890
FACTOR:DISPLAY('K =') REPLY(BUFFER);                                            XF000900
RESPONSE='';                                                                    XF000910
DO K = 1 TO 80;                                                                 XF000920
 IF SUBSTR(BUFFER,K,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,K,1);         XF000930
END;                                                                            XF000940
DO M = 1 TO LENGTH(RESPONSE);                                                   XF000950
 IF INDEX(DIGITS1,SUBSTR(RESPONSE,M,1)) = 0 THEN DO;                            XF000960
```

```
            DISPLAY (RESPONSE);                                                 XF000970
            DISPLAY(ERROR);                                                     XF000980
            GO TO FACTOR;                                                       XF000990
                                                    END;                        XF001000
END;                                                                            XF001010
IF RESPONSE = '' THEN RETURN;                                                   XF001020
AMOUNT=RESPONSE;                                                                XF001030
                                                                                XF001040
  /* PERFORM TRANSLATION AND DISPLAY RESULT */                                  XF001050
                                                                                XF001060
RESPONSE='';                                                                    XF001070
DO J = 0 TO DEGREE;                                                             XF001080
   TERM = 0;                                                                    XF001090
   DO I = 0 TO J;                                                               XF001100
     IF AMOUNT¬=0 THEN M = ABS(AMOUNT)**(J-I);                                  XF001110
          ELSE DO; TRANSP(J)=COEFFS(J);                                         XF001120
                GO TO TAB; END;                                                 XF001130
     IF AMOUNT < 0 THEN DO; IF MOD(J-I,2) = 1 THEN M = -M;END;                  XF001140
     TERM=TERM+(COEFFS(I)*FACT(DEGREE-I)*M)/FACT(J-I);                          XF001150
   END;                                                                         XF001160
   TRANSP(J) = TERM/FACT(DEGREE-J);                                             XF001170
   TAB:PUT STRING(RESPONSE) EDIT(TRANSP(J))(F(8));                              XF001180
   NUMBER='';                                                                   XF001190
   DO N = 1 TO LENGTH(RESPONSE);                                                XF001200
      IF SUBSTR(RESPONSE,N,1)¬=' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,         XF001210
                N,1);                                                           XF001220
   END;                                                                         XF001230
   TEMP=TEMP||NUMBER||',';                                                      XF001240
END;                                                                            XF001250
TEMP=SUBSTR(TEMP,1,LENGTH(TEMP)-1);                                             XF001260
DISPLAY(TEMP);                                                                  XF001270
                                                                                XF001280
END TRANSL;                                                                     XF001290
                                                                                XF001300
                                                                                XF001310
DILATE:PROCEDURE;                                                               XF001320
                                                                                XF001330
  /* THE DILATE PROCEDURE PERFORMS THE DILATION TRANSFORMATION */               XF001340
                                                                                XF001350
  /* DETERMINE DILATION FACTOR */                                               XF001360
                                                                                XF001370
TEMP = '';                                                                      XF001380
HEAD:DISPLAY('A= "INTEGER" OR "RECIPROCAL"?') REPLY(CHOICE);                    XF001390
IF CHOICE = '' THEN DO; DISPLAY (ERROR);                                        XF001400
                        GO TO HEAD;                                             XF001410
                        END;                                                    XF001420
CHOICE = SUBSTR(CHOICE,1,1);                                                    XF001430
IF CHOICE¬='I' & CHOICE ¬='R' THEN DO;                                           XF001440
```

```
        DISPLAY(CHOICE);                                                    XF001450
        DISPLAY(ERROR);                                                     XF001460
        GO TO HEAD;                                                         XF001470
        END;                                                                XF001480
DISPLAY('A =') REPLY(BUFFER);                                               XF001490
RESPONSE = '';                                                              XF001500
DO L = 1 TO 80;                                                             XF001510
   IF SUBSTR(BUFFER,L,1) ¬= ' ' THEN RESPONSE = RESPONSE||SUBSTR(BUFFER     XF001520
      ,L,1);                                                                XF001530
END;                                                                        XF001540
DO L = 1 TO LENGTH(RESPONSE);                                               XF001550
   IF INDEX(DIGITS1,SUBSTR(RESPONSE,L,1))=0 THEN DO;                        XF001560
        DISPLAY(RESPONSE);                                                  XF001570
        DISPLAY(ERROR);                                                     XF001580
        GO TO HEAD;                                                         XF001590
        END;                                                                XF001600
END;                                                                        XF001610
IF RESPONSE = '' THEN RETURN;                                               XF001620
AMOUNT=RESPONSE;                                                            XF001630
                                                                            XF001640
  /* PERFORM THE DILATION TRANSFORMATION AND DISPLAY THE RESULT */          XF001650
                                                                            XF001660
IF AMOUNT = 0 THEN DO; DISPLAY('DILATION FACTOR OF ZERO IS UNDEFINED');     XF001670
                    GO TO HEAD; END;                                        XF001680
DO K = 0 TO DEGREE;                                                         XF001690
    M = ABS(AMOUNT)**(K);                                                   XF001700
    IF AMOUNT < 0 THEN DO: IF MOD(K,2)=1 THEN M = -M; END;                  XF001710
    IF CHOICE = 'I' THEN TRANSP(K)=COEFFS(K)*M;                             XF001720
       ELSE DO; IF MOD(COEFFS(K),M)=0 THEN TRANSP(K)=COEFFS(K)/M;           XF001730
                    ELSE DO;DISPLAY('NON-INTEGER COEFFICIENTS');            XF001740
                         GO TO ASK;                                         XF001750
                       END;                                                 XF001760
            END;                                                            XF001770
    PUT STRING (RESPONSE) EDIT (TRANSP(K))(F(8));                           XF001780
    NUMBER='';                                                              XF001790
    DO N=1 TO LENGTH(RESPONSE);                                             XF001800
      IF SUBSTR(RESPONSE,N,1)¬=' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,     XF001810
                  N,1);                                                     XF001820
    END;                                                                    XF001830
    TEMP=TEMP||NUMBER||',';                                                 XF001840
END;                                                                        XF001850
TEMP=SUBSTR(TEMP,1,LENGTH(TEMP)-1);                                         XF001860
DISPLAY(TEMP);                                                              XF001870
                                                                            XF001880
END DILATE;                                                                 XF001890
                                                                            XF001900
MONIC:PROCEDURE;                                                            XF001910
                                                                            XF001920
```

```
   /* MONIC TRANSFORMS A GIVEN POLYNOMIAL INTO A MONIC POLYNOMIAL      XF001930
         VIA THE M TRANSFORMATION.  */                                 XF001940
                                                                       XF001950
TEMP='';                                                               XF001960
AMOUNT=COEFFS(0);                                                      XF001970
IF AMOUNT = 0 THEN DO;                                                 XF001980
       DISPLAY('TRANSFORMATION IMPOSSIBLE-LEADING COEFFICIENT ZERO');  XF001990
       RETURN;                                                         XF002000
       END;                                                            XF002010
TRANSP(0)=1;                                                           XF002020
TRANSP(1) = COEFFS(1);                                                 XF002030
DO K = 2 TO DEGREE;                                                    XF002040
    M=ABS(AMOUNT)**(K-1);                                              XF002050
    IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M=-M;                            XF002060
    TRANSP(K)=COEFFS(K)*M;                                             XF002070
    PUT STRING(RESPONSE) EDIT(TRANSP(K))(F(15));                       XF002080
    NUMBER = '';                                                       XF002090
    DO N = 1 TO LENGTH(RESPONSE);                                      XF002100
      IF SUBSTR(RESPONSE,N,1)¬=' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,XF002110
            N,1);                                                      XF002120
    END;                                                               XF002130
    TEMP=TEMP||NUMBER||',';                                            XF002140
END;                                                                   XF002150
NUMBER = '';                                                           XF002160
DO N = 1 TO LENGTH(CHAR(TRANSP(0)));                                   XF002170
    IF SUBSTR(CHAR(TRANSP(0)),N,1)¬=' ' THEN NUMBER=NUMBER||SUBSTR(    XF002180
                        CHAR(TRANSP(0)),N,1);                          XF002190
  END;                                                                 XF002200
NUMBER=NUMBER||',';                                                    XF002210
DO N=1 TO LENGTH(CHAR(TRANSP(1)));                                     XF002220
  IF SUBSTR(CHAR(TRANSP(1)),N,1)¬=' ' THEN NUMBER = NUMBER||SUBSTR(    XF002230
                CHAR(TRANSP(1)),N,1);                                  XF002240
END;                                                                   XF002250
TEMP = NUMBER||','||SUBSTR(TEMP,1,LENGTH(TEMP)-1);                     XF002260
DISPLAY(TEMP);                                                         XF002270
END MONIC;                                                             XF002280
                                                                       XF002290
FACT:PROCEDURE (N) FIXED BIN (31) RECURSIVE;                           XF002300
                                                                       XF002310
  /* THE FACT PROCEDURE COMPUTES N FACTORIAL FOR A GIVEN N */          XF002320
                                                                       XF002330
   DCL (F,N) FIXED BIN (31);                                           XF002340
   IF N ¬= 0 THEN GO TO SKIP;                                          XF002350
   F=1;                                                                XF002360
   RETURN (F);                                                         XF002370
   SKIP:N=N-1;                                                         XF002380
   F=FACT(N);                                                          XF002390
   N=N+1;                                                              XF002400
```

```
   F=F*N;                                                                    XF002410
   RETURN (F);                                                               XF002420
   END FACT;                                                                 XF002430
                                                                             XF002440
  /* DETERMINE NEXT PROGRAM OPTION */                                        XF002450
                                                                             XF002460
  ASK:DISPLAY('NEXT OPTION?') REPLY(RESPONSE);                               XF002470
   IF RESPONSE='' THEN GO TO TOP;                                            XF002480
   RESPONSE=SUBSTR(RESPONSE,1,1);                                            XF002490
   IF RESPONSE = 'R' THEN DO; COEFFS=TRANSP;                                 XF002500
                              GO TO TRANS;                                   XF002510
                              END;                                           XF002520
   IF RESPONSE='P' THEN GO TO TRANS;                                         XF002530
   IF RESPONSE = 'O' THEN DO; COEFFS=POLY; GO TO TRANS; END;                 XF002540
   IF RESPONSE¬='R' & RESPONSE¬='P' & RESPONSE¬='O' THEN DO;                 XF002550
              DISPLAY(ERROR); GO TO ASK; END;                                XF002560
END;      /*END OF BEGIN BLOCK */                                           XF002570
                                                                             XF002580
FINISH:DISPLAY('END OF JOB');                                                XF002590
                                                                             XF002600
END XFORM;                                                                   XF002610
```

NOTES AND INSTRUCTIONS ON USE OF PROGRAM "TESTERA"

1. "TESTERA" is a program which determines the transformational relationship between two primitive polynomials with integer coefficients.

2. The "LIST" data entry specifies whether a single polynomial is to be tested against the first polynomial entered or a succession of polynomials to be tested against the same polynomial. Response is 'Y' (yes) or 'N' (no). If "LIST" is specified, the program returns to the entry point for the second polynomial after completion of testing of each polynomial.

3. The "DEGREE?" is entered as a positive integer.

4. Coefficients of polynomials are entered as integers separated by commas representing coefficients of descending powers of variable. Zero coefficients must be included.

5. Results of tests are displayed as integer values of K and A for transformations between the polynomials or, if no transformational relationship exists, "NO TRANSFORMATION" is displayed.

6. If either or both polynomials are non-monic, they are transformed into monic polynomials prior to testing for relationship between them.

7. Size and implementation restrictions for "TESTERA" are the same as for "XFORM."

```
load testera (xeq)
EXECUTION BEGINS...
LIST?
_no
DEGREE?
_4
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,0,0,54,567
NEXT COEFFICIENTS
_1,0,0,2,7
K=0...A=3***
DEGREE?

_
LIST?
_yes
DEGREE?
_2
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,1,1
NEXT COEFFICIENTS
_1,5,7
K=2...A=1
NEXT COEFFICIENTS
_1,3,9
K=0...A=3
NEXT COEFFICIENTS
_1,-1,1
K=-1...A=1
NEXT COEFFICIENTS
_1,1,2
NO TRANSFORMATIONS
NEXT COEFFICIENTS

_
LIST?
_no
DEGREE?
_12
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,1,1,1,1,1,1,1,1,1,1,1,1
NEXT COEFFICIENTS
_1,2,4,8,16,32,64,128,256,512,1024,2048,4096
K=0...A=2
DEGREE?

_
LIST?

_
END OF JOB
R; T=1.73/4.07 13.40.06
```

```
TESTERA:PROCEDURE OPTIONS(MAIN);                                        TES00010
                                                                        TES00020
    /* TESTERA IS A PROCEDURE WHICH ACCEPTS TWO INTEGRAL POLYNOMIALS   TES00030
       OF EQUAL DEGREE AND DETERMINES THE T AND D TRANSFORMATION        TES00040
       RELATIONS BETWEEN THEM.  */                                      TES00050
                                                                        TES00060
DCL RESPONSE CHAR(512) VARYING INITIAL ('');                            TES00070
DCL BUFFER CHAR(80);                                                    TES00080
DCL DEGREE FIXED BIN (31);                                              TES00090
DCL (TEMP,TEMP1) CHAR(72) VARYING;                                      TES00100
DCL NUMBER CHAR(15) VARYING;                                            TES00110
DCL ANSWER CHAR(15) VARYING;                                            TES00120
DCL (KK1,K1) FIXED BIN(31);                                             TES00130
DCL DIGITS CHAR(10) INITIAL ('0123456789');                             TES00140
DCL DIGITS1 CHAR(11) INITIAL ('0123456789-');                           TES00150
DCL STRING1 CHAR(12) INITIAL ('0123456789-,');                          TES00160
DCL (L,J,K,FLAG) FIXED BIN (31);                                        TES00170
DCL (A1,A2) FIXED BIN (31);                                             TES00180
DCL TERM FIXED BIN (31);                                                TES00190
DCL DISC FIXED BIN (31,5);                                              TES00200
DCL CHECK FIXED BIN (15) ;                                              TES00210
DCL ERROR CHAR(25) INITIAL ('INVALID RESPONSE--REENTER');               TES00220
DCL CHOICE CHAR(4) VARYING INITIAL ('');                                TES00230
                                                                        TES00240
  /* DETERMINE WHETHER INPUT IS INDIVIDUAL PAIRS OR A LIST             TES00250
     TO BE TESTED AGAINST A SINGLE POLYNOMIAL. */                       TES00260
                                                                        TES00270
FIRST:DISPLAY('LIST?') REPLY(CHOICE);                                   TES00280
    IF CHOICE='' THEN GO TO FINISH;                                     TES00290
    CHOICE=SUBSTR(CHOICE,1,1);                                          TES00300
    IF CHOICE¬='Y' & CHOICE¬='N' THEN DO; DISPLAY (CHOICE);             TES00310
                                          DISPLAY(ERROR);               TES00320
                                          GO TO FIRST; END;             TES00330
                                                                        TES00340
    /* DETERMINE DEGREE OF POLYNOMIALS. */                              TES00350
                                                                        TES00360
START:RESPONSE='';                                                      TES00370
    DISPLAY('DEGREE?')REPLY(BUFFER);                                    TES00380
    DO L = 1 TO 80;                                                     TES00390
       IF SUBSTR(BUFFER,L,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,TES00400
          L,1);                                                         TES00410
    END;                                                                TES00420
    IF RESPONSE = '' THEN GO TO FIRST;                                  TES00430
    DO L = 1 TO LENGTH(RESPONSE);                                       TES00440
       IF INDEX(DIGITS,SUBSTR(RESPONSE,L,1))=0 THEN DO;                 TES00450
                          DISPLAY(SUBSTR(RESPONSE,L,1));                TES00460
                          DISPLAY(ERROR);                               TES00470
                          GO TO START;                                  TES00480
```

```
                                END;                                                    TES00490
    END;                                                                                TES00500
    DEGREE = RESPONSE;                                                                  TES00510
BEGIN;                                                                                  TES00520
                                                                                        TES00530
    /* DECLARE ARRAYS OF PROPER SIZE FOR DEGREE. */                                     TES00540
                                                                                        TES00550
  DCL COEFFS(2,0:DEGREE) FIXED BIN (31);                                                TES00560
  DCL QUAD(3) FIXED BIN (31);                                                           TES00570
                                                                                        TES00580
    /* REQUEST COEFFICIENTS OF FIRST POLYNOMIAL AND PROCESS INPUT.*/                    TES00590
                                                                                        TES00600
TOP:DISPLAY('FIRST COEFFICIENTS SEPARATED BY COMMAS')                                   TES00610
            REPLY (BUFFER);                                                             TES00620
  RESPONSE='';                                                                          TES00630
  DO L = 1 TO 80;                                                                       TES00640
    IF SUBSTR(BUFFER,L,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,                   TES00650
            L,1);                                                                       TES00660
  END;                                                                                  TES00670
  DO L = 1 TO LENGTH(RESPONSE);                                                         TES00680
    IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;                                   TES00690
              DISPLAY(SUBSTR(RESPONSE,L,1));                                            TES00700
              DISPLAY(ERROR); GO TO TOP; END;                                           TES00710
  END;                                                                                  TES00720
  DO L = 0 TO DEGREE;                                                                   TES00730
    IF RESPONSE = '' THEN GO TO FIRST;                                                  TES00740
    K=INDEX(RESPONSE,',');                                                              TES00750
    IF K=1 THEN DO; DISPLAY(RESPONSE);DISPLAY(ERROR);GO TO TOP; END;                    TES00760
    IF K=0 THEN K=LENGTH(RESPONSE)+1;                                                   TES00770
    COEFFS(1,L)=SUBSTR(RESPONSE,1,K-1);                                                 TES00780
    IF INDEX(RESPONSE,',')=0 THEN RESPONSE='';                                          TES00790
    IF LENGTH(RESPONSE)>K THEN RESPONSE=SUBSTR(RESPONSE,K+1);                           TES00800
       ELSE RESPONSE='';                                                                TES00810
  END;                                                                                  TES00820
                                                                                        TES00830
    /* NOW REQUEST SECOND SET OF COEFFICIENTS AND PROCESS INPUT. */                     TES00840
                                                                                        TES00850
SECOND:DISPLAY('NEXT COEFFICIENTS') REPLY(BUFFER);                                      TES00860
 RESPONSE='';                                                                           TES00870
 DO L = 1 TO 80;                                                                        TES00880
  IF SUBSTR(BUFFER,L,1)¬=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);                TES00890
 END;                                                                                   TES00900
 DO L = 1 TO LENGTH(RESPONSE);                                                          TES00910
   IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;                                    TES00920
     DISPLAY(SUBSTR(RESPONSE,L,1));DISPLAY(ERROR); GO TO SECOND; END;                   TES00930
 END;                                                                                   TES00940
 DO L = 0 TO DEGREE;                                                                    TES00950
    IF RESPONSE='' THEN GO TO FIRST;                                                    TES00960
```

```
   K=INDEX(RESPONSE,',');                                                    TES00970
   IF K=1 THEN DO; DISPLAY(RESPONSE); DISPLAY(ERROR); GO TO SECOND;END;      TES00980
   IF K=0 THEN K=LENGTH(RESPONSE)+1;                                         TES00990
   COEFFS(2,L)=SUBSTR(RESPONSE,1,K-1);                                       TES01000
   IF INDEX(RESPONSE,',')=0 THEN RESPONSE='';                                TES01010
   IF LENGTH(RESPONSE) > K THEN RESPONSE=SUBSTR(RESPONSE,K+1);               TES01020
      ELSE RESPONSE='';                                                      TES01030
END;                                                                         TES01040
                                                                             TES01050
   /* NOW CHECK FOR PROPER DEGREE OF POLYNOMIAL. */                          TES01060
                                                                             TES01070
IF COEFFS(1,0)=0 THEN DO; DISPLAY('INCORRECT DEGREE-FIRST POLYNOMIAL');      TES01080
                          GO TO START; END;                                  TES01090
IF COEFFS(2,0)=0 THEN DO;DISPLAY('INCORRECT DEGREE-SECOND POLYNOMIAL');      TES01100
                          GO TO START; END;                                  TES01110
                                                                             TES01120
  /* IF POLYNOMIALS ARE NOT MONIC,THEN TRANSFORM THEM TO                    TES01130
     MONIC POLYNOMIALS PRIOR TO TEST FOR TRANSFORMATIONAL                    TES01140
      RELATIONSHIP. */                                                       TES01150
                                                                             TES01160
IF COEFFS(1,0) ¬= 1 THEN DO;FLAG=1;CALL MONIC; END;                          TES01170
IF COEFFS(2,0)¬=1 THEN DO; FLAG=2; CALL MONIC; END;                          TES01180
                                                                             TES01190
   /* IF BOTH TRACES ARE ZERO THEN TRANSFORMTION, IF ANY,                    TES01200
      MUST BE PURELY D TYPE, SO USE SPECIAL TEST FOR THIS                    TES01210
      CASE. */                                                               TES01220
                                                                             TES01230
IF COEFFS(1,1)=0 & COEFFS(2,1)=0 THEN DO;                                    TES01240
  IF MOD(COEFFS(2,DEGREE),COEFFS(1,DEGREE))=0 THEN DO;                       TES01250
     KK1=COEFFS(2,DEGREE)/COEFFS(1,DEGREE);                                  TES01260
     L = 0;                                                                  TES01270
TAB1:L=L+1;                                                                  TES01280
     TERM=1;                                                                 TES01290
     DO J = 1 TO DEGREE;                                                     TES01300
       TERM=TERM*L;                                                          TES01310
     END;                                                                    TES01320
     IF TERM < KK1 THEN GO TO TAB1;                                          TES01330
     IF TERM = KK1 THEN DO;                                                  TES01340
          A1 = L;                                                            TES01350
          K1 = 0;                                                            TES01360
          FLAG = 2;                                                          TES01370
          CHECK = 1;                                                         TES01380
          GO TO GOTONE;                                                      TES01390
          END;                                                               TES01400
        END;                                                                 TES01410
     IF MOD(COEFFS(1,DEGREE),COEFFS(2,DEGREE))=0 THEN DO;                    TES01420
        KK1=COEFFS(1,DEGREE)/COEFFS(2,DEGREE);                               TES01430
        L = 0;                                                               TES01440
```

```
      TAB2:L=L+1;                                                          TES01450
         TERM=1;                                                           TES01460
         DO J = 1 TO DEGREE;                                               TES01470
           TERM = TERM*L;                                                  TES01480
         END;                                                              TES01490
         IF TERM < KK1 THEN GO TO TAB2;                                    TES01500
         IF TERM = KK1 THEN DO;                                            TES01510
             A1 = L;                                                       TES01520
             K1 = 0;                                                       TES01530
             FLAG = 4;                                                     TES01540
             CHECK = 0;                                                    TES01550
             GO TO GOTONE;                                                 TES01560
             END;                                                          TES01570
           END;                                                            TES01580
         DISPLAY('NO TRANSFORMATIONS');                                    TES01590
         IF CHOICE = 'Y' THEN GO TO SECOND;                                TES01600
           ELSE GO TO FIRST;                                               TES01610
        END;                                                               TES01620
        CHECK = 1;                                                         TES01630
                                                                           TES01640
   /* IF COEFFS(1,1)=0 THEN MUST SOLVE QUADRATIC IN A VICE K. */           TES01650
                                                                           TES01660
        IF COEFFS(1,1)=0 THEN DO;                                          TES01670
           IF MOD(COEFFS(2,1),(COEFFS(1,0)*DEGREE))¬=0 THEN GO TO OTHER;   TES01680
           K1 = COEFFS(2,1)/(COEFFS(1,0)*DEGREE);                          TES01690
           DISC=((2*COEFFS(1,0)*DEGREE*COEFFS(2,2))                        TES01700
                    -((COEFFS(2,1)**2)*(DEGREE-1)))/                       TES01710
                    (2*COEFFS(1,2)*COEFFS(1,0)*DEGREE);                    TES01720
           IF DISC < 0 THEN GO TO OTHER;                                   TES01730
           DISC = SQRT(DISC);                                              TES01740
           IF MOD(DISC,1)¬=0 THEN GO TO OTHER;                             TES01750
           A1 = DISC;                                                      TES01760
           FLAG = 1;                                                       TES01770
           GO TO GOTONE;                                                   TES01780
        END;                                                               TES01790
                                                                           TES01800
   /* OTHERWISE, SOLVE STANDARD QUADRATIC IN K. FIRST, CLACULATE          TES01810
      COEFFICIENTS OF QUADRATIC. */                                        TES01820
                                                                           TES01830
QUAD(1)=(COEFFS(1,0)*DEGREE*(COEFFS(1,1)**2)*(1-DEGREE))+                  TES01840
        (2*(COEFFS(1,0)**2)*(DEGREE**2)*COEFFS(1,2));                      TES01850
QUAD(2)=(2*COEFFS(2,1)*DEGREE)*((COEFFS(1,1)**2)-(2*COEFFS(1,2)            TES01860
         *COEFFS(1,0)))-(2*COEFFS(2,1)*(COEFFS(1,1)**2));                  TES01870
QUAD(3)=(2*COEFFS(1,2)*(COEFFS(2,1)**2))-(2*(COEFFS(1,1)**2)*              TES01880
           COEFFS(2,2));                                                   TES01890
                                                                           TES01900
   /* CHECK FOR SPECIAL CASES OF QUADRATIC. */                             TES01910
                                                                           TES01920
```

```
  IF QUAD(1)=0 THEN DO;                                                          TES01930
     IF QUAD(2) = 0 THEN GO TO OTHER;                                            TES01940
          IF MOD(QUAD(3),QUAD(2))¬=0 THEN GO TO OTHER;                           TES01950
          K1=QUAD(3)/QUAD(2);                                                    TES01960
          GO TO L1;                                                              TES01970
          END;                                                                   TES01980
  DISC=(QUAD(2)**2)-(4*QUAD(1)*QUAD(3));                                         TES01990
  IF DISC < 0 THEN GO TO OTHER;                                                  TES02000
  IF MOD(SQRT(DISC),1) ¬= 0 THEN GO TO OTHER;                                    TES02010
  TERM = SQRT(DISC);                                                             TES02020
  KK1 = -QUAD(2)-TERM;                                                           TES02030
  IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NEXT;                                      TES02040
  K1=KK1/(2*QUAD(1));                                                            TES02050
L1:IF MOD(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE),COEFFS(1,1))¬=0 THEN               TES02060
        GO TO NEXT;                                                              TES02070
  A1=(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE))/COEFFS(1,1);                          TES02080
  FLAG=1;                                                                        TES02090
  GO TO GOTONE;                                                                  TES02100
                                                                                 TES02110
     /* TRY OTHER SOLUTION OF QUADRATIC. */                                      TES02120
                                                                                 TES02130
NEXT:IF COEFFS(1,1)=0 THEN DO; A1=DISC;                                          TES02140
                               FLAG=2;                                           TES02150
                               GO TO GOTONE; END;                                TES02160
     IF QUAD(1)=0 THEN GO TO OTHER;                                              TES02170
     KK1=-QUAD(2)+TERM;                                                          TES02180
     IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO OTHER;                                  TES02190
     K1 = KK1 / (2*QUAD(1));                                                     TES02200
     IF MOD(COEFFS(2,1),(DEGREE*K1)+(COEFFS(1,1)))¬= 0 THEN GO TO OTHER;         TES02210
     A1 = COEFFS(2,1) / ((DEGREE*K1)+(COEFFS(1,1)));                             TES02220
     FLAG = 2;                                                                   TES02230
     GO TO GOTONE;                                                               TES02240
                                                                                 TES02250
    /* FLIP POLYNOMIALS TO CHECK FOR TRANSFORMATIONS IN OTHER                   TES02260
       DIRECTION. */                                                             TES02270
                                                                                 TES02280
OTHER:CHECK=0;                                                                   TES02290
                                                                                 TES02300
    /* IF COEFFS(2,1)=0 THEN SOLVE QUADRATIC IN A VICE K. */                    TES02310
                                                                                 TES02320
     IF COEFFS(2,1)=0 THEN DO;                                                   TES02330
       IF MOD(COEFFS(1,1),(COEFFS(2,0)*DEGREE))¬=0 THEN GO TO NONE;              TES02340
       K1 = COEFFS(2,1)/(COEFFS(1,0)*DEGREE);                                    TES02350
       DISC=((2*COEFFS(2,0)*DEGREE*COEFFS(1,2))                                  TES02360
              -((COEFFS(1,1)**2)*(DEGREE-1)))/                                   TES02370
             (2*COEFFS(2,2)*COEFFS(2,0)*DEGREE);                                 TES02380
       IF DISC < 0 THEN GO TO NONE;                                              TES02390
       DISC = SQRT(DISC);                                                        TES02400
```

```
      IF MOD(DISC,1)¬=0 THEN GO TO NONE;                                    TES02410
      A1 = DISC;                                                            TES02420
      FLAG=3;                                                               TES02430
      GO TO GOTONE;                                                         TES02440
     END;                                                                   TES02450
                                                                            TES02460
   /* OTHERWISE, SOLVE STANDARD QUADRATIC IN K. */                          TES02470
 QUAD(1)=(COEFFS(2,0)*DEGREE*(COEFFS(2,1)**2)*(1-DEGREE))                   TES02480
                 +(2*(COEFFS(2,0)**2)*(DEGREE**2)*COEFFS(2,2));             TES02490
 QUAD(2)=(2*COEFFS(1,1)*DEGREE)*((COEFFS(2,1)**2)-(2*COEFFS(2,2)            TES02500
          *COEFFS(2,0)))-(2*COEFFS(1,1)*(COEFFS(2,1)**2));                  TES02510
 QUAD(3)=(2*COEFFS(2,2)*(COEFFS(1,1)**2))-(2*(COEFFS(2,1)**2)               TES02520
           *COEFFS(1,2));                                                   TES02530
                                                                            TES02540
     /* CHECK FOR SPECIAL CONDITIONS IN QUADRATIC. */                       TES02550
                                                                            TES02560
 IF QUAD(1)=0 THEN DO;                                                      TES02570
    IF QUAD(2) = 0 THEN GO TO NONE;                                         TES02580
    IF MOD(QUAD(3),QUAD(2))¬=0 THEN GO TO NONE;                             TES02590
    K1 = QUAD(3)/QUAD(2);                                                   TES02600
    GO TO L2;                                                               TES02610
   END;                                                                     TES02620
 DISC = (QUAD(2)**2)-(4*QUAD(1)*QUAD(3));                                   TES02630
 IF DISC < 0 THEN GO TO NONE;                                               TES02640
 IF MOD(SQRT(DISC),1)¬=0 THEN GO TO NONE;                                   TES02650
 TERM = SQRT(DISC);                                                         TES02660
 KK1 = -QUAD(2)-TERM;                                                       TES02670
 IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NEXT1;                                 TES02680
 K1 = KK1/(2*QUAD(1));                                                      TES02690
L2:IF MOD(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE),COEFFS(2,1))¬=0 THEN          TES02700
      GO TO NEXT1;                                                          TES02710
 A1=(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE))/COEFFS(2,1);                      TES02720
 FLAG=3;                                                                    TES02730
 GO TO GOTONE;                                                              TES02740
                                                                            TES02750
   /* TRY OTHER SOLUTION OF  QUADRATIC. */                                  TES02760
                                                                            TES02770
NEXT1:IF COEFFS(2,1) = 0 THEN DO; A1 = -DISC;                               TES02780
                                  FLAG=4; GO TO GOTONE; END;                TES02790
 IF QUAD(1) = 0 THEN GO TO NONE;                                            TES02800
 KK1=-QUAD(2)+TERM;                                                         TES02810
 IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NONE;                                  TES02820
 K1 = KK1 / (2*QUAD(1));                                                    TES02830
 IF MOD(COEFFS(1,1),(DEGREE*K1)+(COEFFS(2,1)))¬=0 THEN GO TO NONE;          TES02840
 A1=COEFFS(1,1)/((DEGREE*K1)+(COEFFS(2,1)));                                TES02850
 FLAG = 4;                                                                  TES02860
 GO TO GOTONE;                                                              TES02870
                                                                            TES02880
```

```
   /* IF NO TRANSFORMATIONS, INFORM OPERATOR. */                          TES02890
                                                                          TES02900
NONE:DISPLAY('NO TRANSFORMATIONS');                                       TES02910
     IF CHOICE='Y' THEN GO TO SECOND;                                     TES02920
        ELSE GO TO START;                                                 TES02930
                                                                          TES02940
   /* IF POSIBLE VALUE FOR K AND A FOUND, THEN CALCULATE TRIAL           TES02950
      POLYNOMIAL FOR COMPARISON. */                                       TES02960
                                                                          TES02970
                                                                          TES02980
GOTONE:CALL TRANSFO;                                                      TES02990
                                                                          TES03000
   /* PROCESS RESULT OF COMPARISON. */                                    TES03010
                                                                          TES03020
  /* IF COMPARISON CHECKS, THEN PRINT RESULT AND K AND A. */              TES03030
                                                                          TES03040
 IF ANSWER='YES' THEN DO;                                                 TES03050
      PUT STRING(TEMP) EDIT ('K=',K1,'...A=',A1)(A,F(8),A,F(8));          TES03060
      TEMP1='';                                                           TES03070
      DO K=1 TO LENGTH(TEMP);                                             TES03080
        IF SUBSTR(TEMP,K,1)¬=' ' THEN TEMP1=TEMP1||SUBSTR(TEMP,           TES03090
                  K,1);                                                   TES03100
      END;                                                                TES03110
      IF CHECK=1 THEN DISPLAY(TEMP1);                                     TES03120
           ELSE DISPLAY(TEMP1||'***');                                    TES03130
                                                                          TES03140
    /* RETURN FOR NEXT POLYNOMIAL. */                                     TES03150
                                                                          TES03160
      IF CHOICE='Y' THEN GO TO SECOND;                                    TES03170
           ELSE GO TO START;                                              TES03180
      END;                                                                TES03190
                                                                          TES03200
   /* IF COMPARISON FAILS, THEN RETURN TO NEXT POINT OF CALCULATION. */   TES03210
                                                                          TES03220
   ELSE DO; IF FLAG = 1 THEN GO TO NEXT;                                  TES03230
            IF FLAG = 2 THEN GO TO OTHER;                                 TES03240
            IF FLAG = 3 THEN GO TO NEXT1;                                 TES03250
            IF FLAG = 4 THEN GO TO NONE;                                  TES03260
        END;                                                              TES03270
                                                                          TES03280
DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));                 TES03290
                                                                          TES03300
FACT:PROCEDURE (N) FIXED BIN (31) RECURSIVE;                              TES03310
                                                                          TES03320
     /* FACT COMPUTES N FACTORIAL */                                      TES03330
                                                                          TES03340
  DCL (F,N) FIXED BIN (31);                                               TES03350
      IF N ¬= 0 THEN GO TO SKIP;                                          TES03360
```

```
      F = 1;                                                                    TES03370
      RETURN (F);                                                               TES03380
 SKIP:N=N-1;                                                                    TES03390
      F = FACT(N);                                                              TES03400
      N = N+1;                                                                  TES03410
      F = F*N;                                                                  TES03420
      RETURN (F);                                                               TES03430
END FACT;                                                                       TES03440
                                                                                TES03450
TRANSFO:PROCEDURE;                                                              TES03460
                                                                                TES03470
     /*TRANSFO PERFORMS THE TRIAL TRANSFORMATIONS AND TESTS THE RESULT          TES03480
         AGAINST THE OTHER POLYNOMIAL */                                        TES03490
 DCL (J,K,L,M,TERM3) FIXED BIN(31);                                             TES03500
 DCL (BASE(0:DEGREE),TRIAL1(0:DEGREE),TRIAL2(0:DEGREE),TEST(0:DEGREE))          TES03510
       FIXED BIN (31);                                                          TES03520
 IF A1 = 0 THEN DO;                                                             TES03530
    ANSWER = 'NO';                                                              TES03540
    RETURN; END;                                                                TES03550
                                                                                TES03560
 IF FLAG = 1 | FLAG = 2 THEN DO; DO L = 0 TO DEGREE;                           TES03570
           BASE(L)=COEFFS(1,L);                                                 TES03580
           TEST(L)=COEFFS(2,L);                                                 TES03590
           END; END;                                                            TES03600
 IF FLAG = 3 | FLAG = 4 THEN DO; DO L = 0 TO DEGREE;                           TES03610
           BASE(L)=COEFFS(2,L);                                                 TES03620
           TEST(L)=COEFFS(1,L);                                                 TES03630
           END; END;                                                            TES03640
                                                                                TES03650
    /* PERFORM D TRANSFORMATION. */                                             TES03660
                                                                                TES03670
 DO K = 0 TO DEGREE;                                                            TES03680
    M = ABS(A1)**K;                                                             TES03690
    IF A1 < 0 THEN DO; IF MOD(K,2)=1 THEN M = -M; END;                          TES03700
    TRIAL1(K) = BASE(K)*M;                                                      TES03710
 END;                                                                           TES03720
                                                                                TES03730
    /* PERFORM T TRANSFORMATION. */                                             TES03740
                                                                                TES03750
 DO J = 0 TO DEGREE;                                                            TES03760
     TERM3 = 0;                                                                 TES03770
     DO I = 0 TO J;                                                             TES03780
       IF K1 ¬= 0 THEN M=ABS(K1)**(J-I);                                        TES03790
          ELSE DO; TRIAL2(J) = TRIAL1(J);                                       TES03800
                   GO TO TAB; END;                                              TES03810
       IF K1 < 0 THEN DO; IF MOD(J-I,2)=1 THEN M=-M; END;                       TES03820
       TERM3 = TERM3 + (TRIAL1(I)*FACT(DEGREE-I)*M)/FACT(J-I);                  TES03830
     END;                                                                       TES03840
```

```
      TRIAL2(J) = TERM3/FACT(DEGREE-J);                                  TES03850
  TAB:END;                                                               TES03860
  ANSWER='YES';                                                          TES03870
                                                                         TES03880
    /* TEST RESULT. */                                                   TES03890
                                                                         TES03900
  DO J = 0 TO DEGREE;                                                    TES03910
     IF TRIAL2(J)¬=TEST(J) THEN DO; ANSWER='NO'; RETURN; END;            TES03920
  END;                                                                   TES03930
                                                                         TES03940
  END TRANSFO;                                                           TES03950
                                                                         TES03960
 MONIC:PROCEDURE;                                                        TES03970
                                                                         TES03980
     /* MONIC TRANSFORMS A GIVEN POLYNOMIAL INTO A MONIC VIA            TES03990
         THE M TRANSFORMATION.  */                                       TES04000
                                                                         TES04010
  DCL (M,K,L,AMCUNT) FIXED BIN (31);                                     TES04020
  DCL HOLD(0:DEGREE) FIXED BIN (31);                                     TES04030
                                                                         TES04040
  AMOUNT = COEFFS(FLAG,0);                                               TES04050
  HOLD(0)=1;                                                             TES04060
  HOLD(1)=COEFFS(FLAG,1);                                                TES04070
  DO K = 2 TO DEGREE;                                                    TES04080
     M=ABS(AMOUNT)**(K-1);                                               TES04090
     IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M = -M;                           TES04100
     HOLD(K)= COEFFS(FLAG,K)*M;                                          TES04110
  END;                                                                   TES04120
  TEMP ='';                                                              TES04130
  DO K = 0 TO DEGREE;                                                    TES04140
     COEFFS(FLAG,K) = HOLD(K);                                           TES04150
     PUT STRING(RESPONSE) EDIT(HOLD(K))(F(15));                          TES04160
     NUMBER = '';                                                        TES04170
     DO L = 1 TO LENGTH(RESPONSE);                                       TES04180
        IF SUBSTR(RESPONSE,L,1)¬=' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,TES04190
             L,1);                                                       TES04200
     END;                                                                TES04210
     TEMP = TEMP||NUMBER||',';                                           TES04220
  END;                                                                   TES04230
  TEMP = SUBSTR(TEMP,1,LENGTH(TEMP)-1);                                  TES04240
  PUT STRING(RESPONSE) EDIT('MONIC FOR POLY NUMBER',FLAG,' IS:')         TES04250
     (A,F(1),A);                                                         TES04260
  DISPLAY(RESPONSE);                                                     TES04270
  DISPLAY(TEMP);                                                         TES04280
                                                                         TES04290
 END MONIC;                                                              TES04300
                                                                         TES04310
 END;     /* END OF BEGIN BLOCK */                                       TES04320
```

```
 FINISH:DISPLAY('END OF JOB');                                    TES04330
END TESTERA;                                                      TES04340
```

```
LUTZ2:PROC OPTIONS (MAIN);

    /* LUTZ2 IS A PROGRAM WHICH ACCEPTS A SET OF FUNCTIONAL
       VALUES AND CALCULATES THE LAGRANGIAN POLYNOMIAL
       THEM AND TESTS IT FOR MONICITY AND INTEGRAL
       COEFFICIENTS. */

ON ENDFILE (SYSIN) GO TO FINIS;
DCL DEG FLOAT BIN (53);
DCL (X,Y) FLOAT BIN (53) INITIAL (0);

    /* DETERMINE DEGREE. */

GET LIST (DEG);
BEGIN;

   /* SET UP NECESSARY ARRAYS FOR DEGREE SPECIFIED. */

DCL FVAL(0:DEG) FLOAT BIN (53);
DCL MPWIC(0:DEG) FLOAT BIN (53);

  /* GET SET OF FUNCTIONAL VALUES. */

START:DO K = 0 TO DEG;
   GET LIST (FVAL(K));
END;

   /* CALL PROCEDURE TO CALCULATE LAGRANGIAN POLYNOMIAL. */

CALL LAGRANGE(DEG, FVAL);

  /* TEST FOR MONICITY/ */

IF ABS(MPWIC(0)) = 1 THEN DO; CALL PRINTER;
                                Y = Y+1;
                           END;

   /* INCREMENT COUNTER. */

X = X+1;

   /*RETURN FOR NEXT SET OF VALUES. */

GO TO START;


  LAGRANGE:PROCEDURE (N,ABCISSAE);
```

```
/*LAGRANGE IS A PROCEDURE WHICH CALCULATES THE
   POLYNOMIAL AND TESTS IT FOR INTEGRAL COEFFICIENTS. */

        DCL (J,K,L,M,N,DENOM) FLOAT BIN(53);
        DCL (ABCISSAE(0:N),TERMS(0:N,0:N), ORDINATES(0:N),
             FACTORS(N), POLY(0:N), DENOMS(0:N)) FLOAT BIN(53);

/* LOAD ARRAYS. */

    DO K = 0 TO N;
        MPWIC(K) = 0;
        ORDINATES(K) = K;
    END;

/* CALCULATE POLYNOMIAL. */

    DO J = 0 TO N;

        DENOM = 1;
        DO L = 0 TO J-1, J+1 TO N;
            DENOM = DENOM * (ORDINATES(J) - ORDINATES(L));
        END;
            DENOMS(J) = DENOM;

        M = 0;
        DO L = 0 TO J-1, J+1 TO N;
            M = M + 1;
                FACTORS(M) = (-ORDINATES(L));
        END;
            POLY(N) = FACTORS(N);
            POLY(N-1) = 1;
        DO L = N-2 TO 0 BY -1;
            POLY(L) = 1;
            DO K = 1 TO (N-L-1);
                POLY(L+K) = POLY(L+K+1) + FACTORS(L+1)*POLY(L+K);
            END;
            POLY(N) = POLY(N)*FACTORS(L+1);
        END;
        POLY = POLY * ABCISSAE(J);
        TERMS(J,*) = POLY;
END;
   DO L = 0 TO N;
        DO K = 0 TO L-1, L+1 TO N;
            TERMS(L,*) = TERMS(L,*)*DENOMS(K);
        END;
    END;
    DO L = 0 TO N;
```

```
            DO K = 1 TO N;
                TERMS(0,L) = TERMS(0,L)+TERMS(K,L);
            END;
        END;
        DO L = 1 TO N;
            DENOMS(0) = DENOMS(0) * DENOMS(L);
        END;
        DO L = 0 TO N;
            IF MOD(TERMS(0,L), DENOMS(0)) = 0 THEN
               MPWIC(L) = TERMS(0,L)/DENOMS(0);
                      ELSE DO; DO K = 0 TO N;
                                  MPWIC(K)=0;
                               END;
                               RETURN;
                           END;
        END;
    END LAGRANGE;


PRINTER:PROCEDURE;

    /* PRINTER PRINTS THE POLYNOMIAL OR, IF THE FIRST
       COEFFICIENT IS -1, IT PRINTS THE REFLECTION. */

    IF MPWIC(0) = 1 THEN DO; PUT EDIT('X',DEG)(SKIP,A,F(1,0));
                             DO P = 1 TO DEG-2;
                               PUT EDIT(' ', MPWIC(P),' X',DEG-P)
                                   (A,F(4,0),A,F(1,0));
                               END;
                               PUT EDIT(' ',MPWIC(DEG-1),' X ',
                                    MPWIC(DEG))(A,F(4,0),A,F(4,0));
                             DO P = 0 TO DEG;
                               PUT EDIT('  ',FVAL(P))(A,F(2));
                             END;
                         END;
                    ELSE DO; PUT EDIT('X',DEG)(SKIP,A,F(1,0));
                             DO P = 1 TO DEG-2;
                               PUT EDIT(' ', -MPWIC(P),' X',DEG-P)
                                   (A,F(4,0),A,F(1,0));
                               END;
                               PUT EDIT(' ', -MPWIC(DEG-1),' X ',
                                    -MPWIC(DEG))(A,F(4,0),A,F(4,0));
                             DO P = 0 TO DEG;
                               PUT EDIT('  ', -FVAL(P))(A,F(2));
                             END;
                    END;
END PRINTER;
```

```
END;     /* END OF BEGIN BLOCK. */
FINIS:PUT EDIT('CANDIDATES=',X,'    MPWICS=',Y)(SKIP,A,F(4),A,F(4));
END LUTZ2;
```

```
TESTER:PROCEDURE OPTIONS(MAIN);

 /* TESTER IS A PROGRAM WHICH SEPARATES A DATA SET OF MIPWIC'S
    INTO EQUIVALENCES CLASSES. */
  /* DOCUMENTATION IS SAME AS FOR TESTERA EXCEPT
      AS NOTED. */

 ON ENDFILE (SYSIN) GO TO FINISH;
 DCL RESPONSE CHAR(512) VARYING INITIAL ('');
 DCL BUFFER CHAR(80);
 DCL DEGREE FIXED BIN (31);
 DCL (TEMP,TEMP1) CHAR(72) VARYING;
 DCL NUMBER CHAR(15) VARYING;
 DCL ANSWER CHAR(15) VARYING;
 DCL (KK1,K1) FIXED BIN(31);
 DCL DIGITS CHAR(10) INITIAL ('0123456789');
 DCL DIGITS1 CHAR(11) INITIAL ('0123456789-');
 DCL STRING1 CHAR(12) INITIAL ('0123456789-,');
 DCL (L,J,K,FLAG) FIXED BIN (31);
 DCL (A1,A2) FIXED BIN (31);
 DCL DISC FIXED BIN (31,5);
 DCL TERM FIXED BIN (31);
 DCL CHECK FIXED BIN (15) ;
 DCL ERROR CHAR(25) INITIAL ('INVALID RESPONSE--REENTER');
 DCL CHOICE CHAR(4) VARYING INITIAL ('');
 DCL (SET,INDEX1, INDEX2 ) FIXED BIN (31);

   /* GET DEGREE AND NUMBER OF POLYNOMIALS IN DATA SET. */

 START:GET LIST (DEGREE,SET);
   BEGIN;
      /* DECLARE ARRAYS TO PROPER SIZE. */
      DCL COEFFS(2,0:DEGREE) FIXED BIN (31);
      DCL QUAD(3) FIXED BIN (31);
      DCL DATA(SET,0:DEGREE+1) FIXED BIN (31);
  /* GET FIRST POLYNOMIAL FOR TESTING. */
      DO K = 1 TO SET;
         DO J = 0 TO DEGREE;
            GET LIST (DATA(K,J));
         END;
         DATA(K,DEGREE+1)=1;
      END;
  /* FIND  NEXT POLYNOMIAL FOR TESTING AGAINST FIRST. */
      INDEX1=0;
TOP1:INDEX1=INDEX1+1;
      IF INDEX1=SET+1 THEN GO TO START;
      IF DATA(INDEX1,DEGREE+1)=0 THEN GO TO TOP1;
      DO K = 0 TO DEGREE;
```

```
        COEFFS(1,K)=DATA(INDEX1,K);
      END;
      PUT EDIT('****')(SKIP,A);
      RESPONSE = '';
      DO K = 0 TO DEGREE;
        PUT STRING (TEMP1) EDIT (COEFFS(1,K))(F(8));
        RESPONSE=RESPONSE||TEMP1;
      END;
      DO K = 2 TO LENGTH(RESPONSE);
        IF SUBSTR(RESPONSE,K,1)=' ' THEN RESPONSE=SUBSTR(RESPONSE,
             1,K-1)||SUBSTR(RESPONSE,K+1);
      END;
      IF SUBSTR(RESPONSE,1,1)=' ' THEN RESPONSE=SUBSTR(RESPONSE,2);
      PUT EDIT (RESPONSE)(SKIP,A);
      INDEX2=INDEX1;
TOP2: INDEX2=INDEX2+1;
      IF INDEX2=SET+1 THEN GO TO TOP1;
      IF DATA(INDEX2,DEGREE+1)=0 THEN GO TO TOP2;
      DO K = 0 TO DEGREE;
        COEFFS(2,K)=DATA(INDEX2,K);
      END;
 IF COEFFS(1,0) ¬= 1 THEN DO;FLAG=1;CALL MONIC; END;
 IF COEFFS(2,0)¬=1 THEN DO; FLAG=2; CALL MONIC; END;
 IF COEFFS(1,1)=0 & COEFFS(2,1)=0 THEN DO;
      IF MOD(COEFFS(2,DEGREE),COEFFS(1,DEGREE))=0 THEN DO;
      KK1=COEFFS(2,DEGREE)/COEFFS(1,DEGREE);
      L = 0;
 TAB1:L=L+1;
      TERM=1;
      DO J = 1 TO DEGREE;
        TERM=TERM*L;
      END;
      IF TERM < KK1 THEN GO TO TAB1;
      IF TERM = KK1 THEN DO;
           A1=L;
           K1=0;
           FLAG=2;
           CHECK=1;
           GO TO GOTONE;
         END;
        END;
      IF MOD(COEFFS(1,DEGREE),COEFFS(2,DEGREE))=0 THEN DO;
      KK1=COEFFS(1,DEGREE)/COEFFS(2,DEGREE);
      L=0;
 TAB2:L=L+1;
      TERM=1;
      DO J = 1 TO DEGREE;
        TERM=TERM*L;
```

```
        END;
        IF TERM < KK1 THEN GO TO TAB2;
        IF TERM = KK1 THEN DO;
           A1=L;
           K1=0;
           FLAG=4;
           CHECK=0;
           GO TO GOTONE;
         END;
        END;
        GO TO TOP2;
        END;
        CHECK = 1;
  IF COEFFS(1,1)=0 THEN DO;
     IF MOD(COEFFS(2,1),(COEFFS(1,0)*DEGREE))¬=0 THEN GO TO OTHER;
     K1=COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
     DISC=      ((2*COEFFS(1,0)*DEGREE*COEFFS(2,2))
       -((COEFFS(2,1)**2)*(DEGREE-1)))/
       (2*COEFFS(1,2)*COEFFS(1,0)*DEGREE) ;
     IF DISC < 0 THEN GO TO OTHER;
     DISC=SQRT(DISC);
     IF MOD(DISC,1)¬=0 THEN GO TO OTHER;
     A1=DISC;
     FLAG=1;
     GO TO GOTONE;
     END;
  QUAD(1)=(COEFFS(1,0)*DEGREE*(COEFFS(1,1)**2)*(1-DEGREE))+
           (2*(COEFFS(1,0)**2)*(DEGREE**2)*COEFFS(1,2));
  QUAD(2)=(2*COEFFS(2,1)*DEGREE)*((COEFFS(1,1)**2)-(2*COEFFS(1,2)
            *COEFFS(1,0)))-(2*COEFFS(2,1)*(COEFFS(1,1)**2));
  QUAD(3)=(2*COEFFS(1,2)*(COEFFS(2,1)**2))-(2*(COEFFS(1,1)**2)*
            COEFFS(2,2));
   IF QUAD(1)=0 THEN DO;
    IF QUAD(2)=0 THEN GO TO OTHER;
    IF MOD(QUAD(3),QUAD(2))¬=0 THEN GO TO OTHER;
    K1=QUAD(3)/QUAD(2);
    GO TO L1;
     END;
  DISC=(QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
  IF DISC < 0 THEN GO TO OTHER;
  IF MOD(SQRT(DISC),1) ¬= 0 THEN GO TO OTHER;
  TERM = SQRT(DISC);
  KK1 = -QUAD(2)-TERM;
  IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NEXT;
  K1=KK1/(2*QUAD(1));
L1: IF MOD(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE),COEFFS(1,1))¬=0
      THEN GO TO NEXT;
   A1=(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE))/COEFFS(1,1);
```

```
   FLAG=1;
   GO TO GOTONE;
NEXT:IF COEFFS(1,1)=0 THEN DO; A1=-DISC;
                               FLAG=2;
                               GO TO GOTONE;
                            END;
   IF QUAD(1)=0 THEN GO TO OTHER;
   KK1=-QUAD(2)+TERM;
   IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO OTHER;
   K1 = KK1 / (2*QUAD(1));
   IF MOD(COEFFS(2,1),(DEGREE*K1)+(COEFFS(1,1)))¬= 0 THEN GO TO OTHER;
   A1 = COEFFS(2,1) / ((DEGREE*K1)+(COEFFS(1,1)));
   FLAG = 2;
   GO TO GOTONE;
OTHER:CHECK=0;
   IF COEFFS(2,1)=0 THEN DO;
      IF MOD(COEFFS(1,1),(COEFFS(2,0)*DEGREE))¬=0 THEN GO TO NONE;
      K1=COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
      DISC=     ((2*COEFFS(2,0)*DEGREE*COEFFS(1,2))
        -((COEFFS(1,1)**2)*(DEGREE-1)))/
        (2*COEFFS(2,2)*COEFFS(2,0)*DEGREE) ;
      IF DISC < 0 THEN GO TO NONE;
      DISC = SQRT(DISC);
      IF MOD(DISC,1)¬=0 THEN GO TO NONE;
      A1=DISC;
      FLAG=3;
      GO TO GOTONE;
   END;
 QUAD(1)=(COEFFS(2,0)*DEGREE*(COEFFS(2,1)**2)*(1-DEGREE))
                  +(2*(COEFFS(2,0)**2)*(DEGREE**2)*COEFFS(2,2));
 QUAD(2)=(2*COEFFS(1,1)*DEGREE)*((COEFFS(2,1)**2)-(2*COEFFS(2,2)
          *COEFFS(2,0)))-(2*COEFFS(1,1)*(COEFFS(2,1)**2));
 QUAD(3)=(2*COEFFS(2,2)*(COEFFS(1,1)**2))-(2*(COEFFS(2,1)**2)
           *COEFFS(1,2));
 IF QUAD(1)=0 THEN DO;
    IF QUAD(2)=0 THEN GO TO NONE;
    IF MOD(QUAD(3),QUAD(2))¬=0 THEN GO TO NONE;
    K1=QUAD(3)/QUAD(2);
    GO TO L2;
   END;
 DISC = (QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
 IF DISC < 0 THEN GO TO NONE;
 IF MOD(SQRT(DISC),1)¬=0 THEN GO TO NONE;
 TERM = SQRT(DISC);
 KK1 = -QUAD(2)-TERM;
 IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NEXT1;
 K1 = KK1/(2*QUAD(1));
L2:IF MOD(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE),COEFFS(2,1))¬=0 THEN
```

```
     GO TO NEXT1;
  A1=(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE))/COEFFS(2,1);
  FLAG=3;
  GO TO GOTONE;
NEXT1:IF COEFFS(2,1)=0 THEN DO; A1=-DISC;
                                FLAG=4;
                                GO TO GOTONE;
                              END;
  IF QUAD(1)=0 THEN GO TO NONE;
  KK1=-QUAD(2)+TERM;
  IF MOD(KK1,2*QUAD(1))¬=0 THEN GO TO NONE;
  K1 = KK1 / (2*QUAD(1));
  IF MOD(COEFFS(1,1),(DEGREE*K1)+(COEFFS(2,1)))¬=0 THEN GO TO NONE;
  A1=COEFFS(1,1)/((DEGREE*K1)+(COEFFS(2,1)));
  FLAG = 4;
  GO TO GOTONE;
NONE:GO TO TOP2;
GOTONE:CALL TRANSFO;
     IF ANSWER='YES' THEN DO; RESPONSE = '';
          DO K = 0 TO DEGREE;
               PUT STRING (TEMP) EDIT (COEFFS(2,K))(F(8));
               RESPONSE = RESPONSE||TEMP;
             END;
          DO K = 2 TO LENGTH(RESPONSE);
               IF SUBSTR(RESPONSE,K,1) = ' ' THEN RESPONSE =
                         SUBSTR(RESPONSE,1,K-1)||SUBSTR(RESPONSE,K+1);
          END;
          IF SUBSTR(RESPONSE,1,1) = ' ' THEN RESPONSE=SUBSTR(RESPONSE,
                                                          2);
          PUT STRING (TEMP) EDIT (RESPONSE||'   K=',K1,'     A=',A1)
                    (A,F(5),A,F(5));
          IF CHECK = 1 THEN PUT EDIT (TEMP) (SKIP,A);
            ELSE DO; TEMP=TEMP||'***';
                     PUT EDIT (TEMP)(SKIP,A);
                  END;
          DATA(INDEX2,DEGREE+1)=0;
          GO TO TOP2;
        END;
     ELSE DO; IF FLAG=1 THEN GO TO NEXT;
              IF FLAG = 2 THEN GO TO OTHER;
              IF FLAG = 3 THEN GO TO NEXT1;
              IF FLAG = 4 THEN GO TO NONE;
          END;
  DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));

FACT:PROCEDURE (N) FIXED BIN (31) RECURSIVE;
    DCL (F,N) FIXED BIN (31);
```

```
    IF N ¬= 0 THEN GO TO SKIP;
    F = 1;
    RETURN (F);
 SKIP: N=N-1;
    F = FACT(N);
    N = N+1;
    F = F*N;
    RETURN (F);
 END FACT;

TRANSFO:PROCEDURE;
 DCL (J,K,L,M,TERM3) FIXED BIN(31);
 DCL (BASE(0:DEGREE),TRIAL1(0:DEGREE),TRIAL2(0:DEGREE),TEST(0:DEGREE))
       FIXED BIN (31);
   IF A1=0 THEN DO;
     ANSWER='NO';
     RETURN;
    END;
 IF FLAG = 1 | FLAG = 2 THEN DO; DO L = 0 TO DEGREE;
           BASE(L)=COEFFS(1,L);
           TEST(L)=COEFFS(2,L);
           END; END;
 IF FLAG = 3 | FLAG = 4 THEN DO; DO L = 0 TO DEGREE;
           BASE(L)=COEFFS(2,L);
           TEST(L)=COEFFS(1,L);
           END; END;
 DO K = 0 TO DEGREE;
    M = ABS(A1)**K;
    IF A1 < 0 THEN DO; IF MOD(K,2)=1 THEN M = -M; END;
    TRIAL1(K) = BASE(K)*M;
 END;
 DO J = 0 TO DEGREE;
     TERM3 = 0;
     DO I = 0 TO J;
       IF K1 ¬= 0 THEN M=ABS(K1)**(J-I);
           ELSE DO; TRIAL2(J) = TRIAL1(J);
                    GO TO TAB; END;
       IF K1 < 0 THEN DO; IF MOD(J-I,2)=1 THEN M=-M; END;
      TERM3 = TERM3 + (TRIAL1(I)*FACT(DEGREE-I)*M)/FACT(J-I);
     END;
 TRIAL2(J) = TERM3/FACT(DEGREE-J);
 TAB:END;
 ANSWER='YES';
  DO J = 0 TO DEGREE;
 IF TRIAL2(J)¬=TEST(J) THEN DO; ANSWER='NO'; RETURN; END;
END TRANSFO;
```

```
MONIC:PROCEDURE;
 DCL (M,K,L,AMOUNT) FIXED BIN (31);
 DCL HOLD(0:DEGREE) FIXED BIN (31);
 AMOUNT = COEFFS(FLAG,0);
 HOLD(0)=1;
 HOLD(1)=COEFFS(FLAG,1);
 DO K = 2 TO DEGREE;
    M=ABS(AMOUNT)**(K-1);
    IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M = -M;
     COEFFS(FLAG,K) = COEFFS(FLAG,K) * M;
   END;
END MONIC;


  END;
FINISH:PUT EDIT ('END OF JOB') (SKIP,A);

END TESTER;
```

BIBLIOGRAPHY

1. Dorwart, H. L., "Irreducibility of Polynomials," The American Mathematical Monthly, V. XLII, p. 369-381, June-July 1935.

2. Kelly, J. B., "On Factorization of Polynomials," The American Mathematical Monthly, V. 60, p. 375-379, June-July 1953.

3. Knuth, D. E., The Art of Computer Programming, V. 2, p. 360-444, Addison-Wesley, 1969.

4. Lehmer, D. H., "Computer Technology Applied to the Theory of Numbers," Studies in Mathematics, V. 6, p. 117-151, The Mathematical Association of America, 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor Daniel L. Davis, Code 53Dv Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	LT Robert E. Lutz, USN 409 East Water Street Troy, Ohio 45373	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

AN IMPLEMENTED TRANSFORMATIONAL SCHEME FOR MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master's Thesis, June 1971

5. AUTHOR(S) (First name, middle initial, last name)

Robert E. Lutz, Jr.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1971	82	4

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

A discussion of the problem of the irreducibility of polynomials in the ring of integral polynomials establishes the framework of the research. A transformational scheme is postulated to facilitate investigation of the problem. The coherency of the scheme is detailed and the necessary computational techniques developed.

To determine the efficacy of the transformational scheme, the specification and collection of appropriate sets of data are discussed. The transformational scheme is then applied to the data and the results are tabulated and discussed.

The conclusion is drawn that a transformational scheme is a useful tool for the investigation of the irreducibility of this type of polynomial and some hypotheses as to the refinement and extension of the technique are stated.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Irreducible Polynomials						
Computer-Implemented Transformational Scheme						

Thesis 128470
L93 Lutz
c.1 An implemented trans-
formational scheme for
monic polynomials with
interger coefficients.